سلسلة كن أسدا للإبداع تقدم:

البحر الشاسع لدخول الخوارزميات من بابها الواسع

كل ما يلزمك معرفته لتصبح مبرمجا قويا



خالد السعداني

www.mobarmijoun.com

تقديم: لماذا نبرمج؟

الحمد لله معز الحق وناصره، ومذل الباطل وقاصره، علام الغيوب، ومن بيده أزمة القلوب، الخبير بما تخفي الضمائر، وتكن السرائر، العالم بما تفضي إليه الأمور، وبخائنة الأعين وما تخفي الصدور، والصلاة والسلام على حبيبنا محمد كاشف الغمة عن الأمة، الناطق فيهم بالحكمة، الصادع بالحق، الداعي إلى الصدق.

ثم أما بعد:

كل البرامج التي تستخدمها على حاسوبك لم تأت من فراغ، بل هي نتاج للتركيبة البشرية التي فطر الله عز وجل الناس عليها، والتي تدفعهم باستمرار إلى إيجاد حلول لوضعيات معينة أو ابتكار وسائل للوصول إلى غايات منشودة كلما ألمت بهم حاجة، وهذا دأب الإنسان وحاله مذ أوجده الله جل وعلا، فكما أن الإنسان في أول عهده عانى من قسوة الأجواء والطقوس وتقلباتها بين قر وحر، فدفعته الحاجة إلى أن يتخذ من الكهوف والغيران مساكنا له ليحتمي بها ويلجأ إليها، ودفعته صعوبة صيد الوحوش باليد فاتخذ من جذع الشجر سلاحا له يهوي به على الطرائد، ودفعه حياؤه من الجنس الآخر فاتخذ من جلد الأنعام لباسا له يواري سوأته، ودفعته الحاجة إلى صنع مراكب يمشي بها في البحر فصنع المطرقة وباقي الأدوات، فكذلك شأن الإنسان اليوم، فإنه كلما ألمت به حاجة أو أصابته داهية، فكر مليا كيف يسلم منها في المرة القادمة عبر ابتكار حلول جديدة لم يعرفها أسلافه.

في بريطانيا، في عشرينيات القرن التاسع عشر احتاج التجار إلى وسائل لنقل البضاعة والسلع الكثيرة بين المناطق المتباعدة، فدفعت هذه الحاجة إلى اختراع أول خط للسكك الحديدية بمحركات بخارية، فتطورت الفكرة لتصبح القطارات بالشكل الذي نراها عليه اليوم.

وحينما احتاج الناس إلى نقل الأنباء والأخبار، قاموا بنشر الصحف، ثم اختراع المذياع، ثم التلفاز، ثم الانترنت.

وحينما احتاج الناس للتواصل فيما بينهم، بدؤوا باستعمال الحمام الزاجل، ثم استخدام البريد الورقي، ثم استخدام التلغراف، ثم الأقمار الصناعية وغيرها.

كلنا شاهدنا كيف تتالت الأمواج البحرية الزلزالية "تسونامي" على شرق آسيا، فدفع ذلك اليابانيين إلى إنشاء مباني مضادة للزلازل، وكلنا شاهد كيف عانت اليابان من مشكل استيراد المنتجات الزراعية بسبب انعدام السهول في أراضيها، فدفعها ذلك إلى إنشاء مدرجات فلاحية على الجبال.

الحاجة أم الاختراع، فلولا حاجة الإنسان إلى الشيء لما شغل باله به، وحاجات الإنسان متغيرة وتتزايد باستمرار، والحاجة هي التي تدفع الناس إلى إنشاء برامج.

اسأل نفسك عن كل برنامج نصبته على حاسوبك: لماذا نصبته؟ وسيكون جوابك حتما ومن غير شك هو حاجتك له، فأنت نصبت مكافح الفيروسات anti-virus لحماية حاسوبك من الأضرار المحتملة، ونصبت برنامج ميكروسوفت وورد microsoft word لأنك تحتاج إلى كتابة ومعالجة النصوص وتنسيقها، ونصبت برنامج قارىء الميديا، لأنك تحتاج إلى مشاهدة تسجيلات مرئية، ونصبت متصفح الويب لأنك تحتاج إلى الدخول إلى المواقع.

نفس الحاجة التي دفعتك إلى تحميل البرنامج وتنصيبه دفعت غيرك من مستخدميه، ودفعت قبلكم جميعا فئة من الناس، فقالوا: نحن نحتاج إلى برنامج يقوم بكذا وكذا، فلما برزت الحاجة برزت شركة برمجية وقالت:أنا لها.

لكن ما يهمنا نحن كأفراد نسعى إلى تعلم البرمجة، هو معرفة الطريقة الصحيحة التي بانتهاجنا لها سننشىء برامج قوية وتطبيقات جيدة بالشكل الذي نطمح إليه أو بالشكل المطلوب منا.

وهذا ما سنعرض له خلال كتابنا هذا، وسنحاول قدر المستطاع أن نسلك سبيل البساطة والتفكيك، بعيدين كل البعد عن الصعوبة والتعقيد، لذلك قد أطيل في فصل معين وأسهب فيه فلا تلوموني وتعذلوني فإني ما أسهبت فيه وأطلت حبا في ذلك أو رغبة في استعراض المعارف، كلا وألف كلا، وإنما طبيعة المدروس تستلزم منا شرحه من كل جوانبه لنفهمه فهما شاملا، ولو لخصناه أو اقتضبناه لشوهناه وأسأنا شرحه، فتصل إليكم المعلومة مغلوطة أو غير كاملة.

يتكون هذا الكتاب من ثلاثة أجزاء يكمل بعضها البعض، فالجزء الأول يتناول مفهوم أنظمة الترميز والطرق التي يعالج بها جهاز الحاسوب البيانات والمكونات المادية المتدخلة في العملية لكي يكون المتعلم على دارية بما يحصل على مستوى الجهاز حينما يخاطبه بأوامر برمجية، والجزء الثاني يعرض باختصار كيفية القيام بالعمليات الحسابية الأساسية على البيانات الثنائية، أما الجزء الثالث فهو يتناول الخوارزميات من البداية بأسلوب متدرج وبأمثلة تطبيقية.

خالد السعداني 2013/10/16

ما الذي سأستفيده إن قرأت هذا الكتاب؟

إذا قرأت هذا الكتاب كاملا وبتمعن فإني أضمن لك بعون ا   ما يلي:

1. فهم سليم لأساس البرمجة
2. القدرة على تعلم أية لغة برمجية مهما بلغت صعوبتها وتعقيداتها
3. إمكانية تعلم أكثر من لغة برمجية في وقت وجيز
4. القدرة على تحليل المشكل الواقعي وتأويله برمجيا عبر الخوارزميات
5. بداية قوية وأساس متين لدخول عالم البرمجة

هل أستطيع قراءة جزء من الكتاب فقط؟

ذلك يتوقف على معارفك ومداركك، إن كنت على دراية بأساسيات البرمجة، أو لك خبرة سابقة مع لغة برمجية معينة، فلا مانع من أن تكتفي بقراءة جزء دون جزء.

لكن إن كان هذا أول عهدك بالبرمجة، فأنت مطالب بقراءة الكتاب كاملا، وتطبيق ما جاء فيه من تمارين، والأخذ بما ورد فيه من إرشادات وتعليمات.

هل علي تعلم الانجليزية لكي أصبح مبرمجا؟

تعلم اللغة الانجليزية في وقتنا الراهن شيء مهم جدا لمن يريد أن يواكب تطورات العصر، بيد أن معظم التقنيات والمعارف باتت تعرض بهذه اللغة، لكن دعني أكلمك بصدق، في عالم البرمجة لست مطالبا أن تكون متقنا ومتمكنا من اللغة الانجليزية، غير صحيح بتاتا، ومن قال

لك ذلك فقد كذبك القول أو حدثك عن جهل منه بالبرمجة، أن تكون مبرمجا لا يعني أن تكون أمريكيا أو بريطانيا، بل يلزمك شيء من الجهد وكثير من الرغبة والحب للبرمجة، واللغة لم تكن عائقا في أخذ العلوم ولن تكون كذلك بإذن الله ، لكنها قد تكون مفيدة في بعض الجوانب، لذلك انس موضوع اللغة الانجليزية فنحن سنتعلم البرمجة وليس فنون التواصل :)

أنا فاشل في الرياضيات هل ذلك سيمنعني من تعلم البرمجة؟

الرياضيات هي جزء بسيط من البرمجة وليست كل البرمجة، وتستطيع أن تكون مبرمجا قويا حتى وإن كانت معارفك في الرياضيات متدنية، لذلك لا ترتبك ولا تشغل بالك بهذا، لأنك لن تحتاج الرياضيات إلا في البرامج التي تستلزم منك القيام بعمليات رياضية وعموما لغات البرمجة قد سهلت هذا المجال بشكل رائع، فكل ما ستحتاجه في برامجك من دوال حسابية (سينيس، كوسينيس،...) موجودة مسبقا وتم تجهيزها من قبل الفريق المطور للغة البرمجة.

"يا أيها الذين آمنوا اتقوا اللّه و قولوا قولا سديدا. يصلح لكم أعمالكم و يغفر لكم ذنوبكم ومن يطع اللّه و رسوله فقد فاز فوزا عظيما"

الأحزاب : 70 و 71

الفهرس

الفصل الأول:

أنظمة تمثيل البيانات

جهاز الحاسوب

تعريف وجيز لجهاز الحاسوب / الحاسب

الحاسوب هو جهاز الكتروني مثله مثل باقي الأجهزة الالكترونية (تلفاز، هاتف، جهاز تسجيل،...) يستخدم لتخزين ومعالجة البيانات، وهو يتكون من جزءين لا ثالث لهما، أحدهما آلي Hardware وهو الجانب المادي الذي يضم مكونات الحاسوب التي نراها ونلمسها، أما الجزء الثاني فهو الجزء البرمجي Software وهو الجانب الخفي المسؤول عن تشغيل البرامج والألعاب والملتيميديا من خلال تحكمه في الجانب الآلي Hardware.

يتكون الحاسوب من مجموعة من العناصر المادية نذكرها فيما يلي أهمها:

الذاكرة الرئيسية أو الحية RAM(Random Access Memory):

يمكننا تعريف الذاكرة بأنها مجموعة من الخانات المتتالية والمرقمة عبر عناوين، وكل خانة يمكنها أن تحتوي على بيانات، تتم معالجتها من قبل وحدة المعالجة، كما يمكن للذاكرة أن تقوم بتخزين البرامج (البرنامج هو مجموعة من الأوامر المتسلسلة التي يتم تنفيذها للحصول على نتيجة معينة)، ويتم تمثيل البيانات في الذاكرة على شكل ثنائي عبر متتاليات من الأصفار والآحاد كما سنرى فيما بعد.

كل خانة في الذاكرة مرقمة لكي يسهل الوصول إلى محتواها من قبل وحدة المعالجة، ويسمى هذا الترقيم بالعنونة، أي أن كل خانة لها عنوانها الخاص Address.

ويمكننا تمثيل الذاكرة الرئيسية بهذا الشكل:



الشكل 1 التمثيل الاصطلاحي للذاكرة الرئيسية

وحدة معالجة البيانات Central Processing Unit:

وهو الجزء المهم في الحاسوب، ويعد بمثابة الدماغ المسؤول عن تنفيذ كل عمليات معالجة البيانات المخزنة في الذاكرة.

ويقوم بكل العمليات الحسابية (الجمع، الطرح، الضرب، القسمة) ويقوم أيضا بالعمليات المنطقية مثل مقارنة البيانات.

تقوم وحدة المعالجة بأخذ الأوامر المخزنة في الذاكرة على شكل بيانات، وتبدأ في تنفيذها بدء من أول أمر وانتهاء بآخر أمر وتقوم بإجراء العمليات الحسابية والمنطقية الواردة في البرنامج المخزن، وكلما اقتضى الأمر تقوم بتخزين الناتج في الذاكرة لتستعمله مع أوامر أخرى، وفي ختام تنفيذ البرنامج تقوم وحدة المعالجة بإرسال النتيجة إلى الجهاز الخاص بعرضها (مثلا طباعة نتيجة عملية حسابية في نافذة الكونسول،ستقوم وحدة المعالجة بإرسال النتيجة إلى الشاشة)

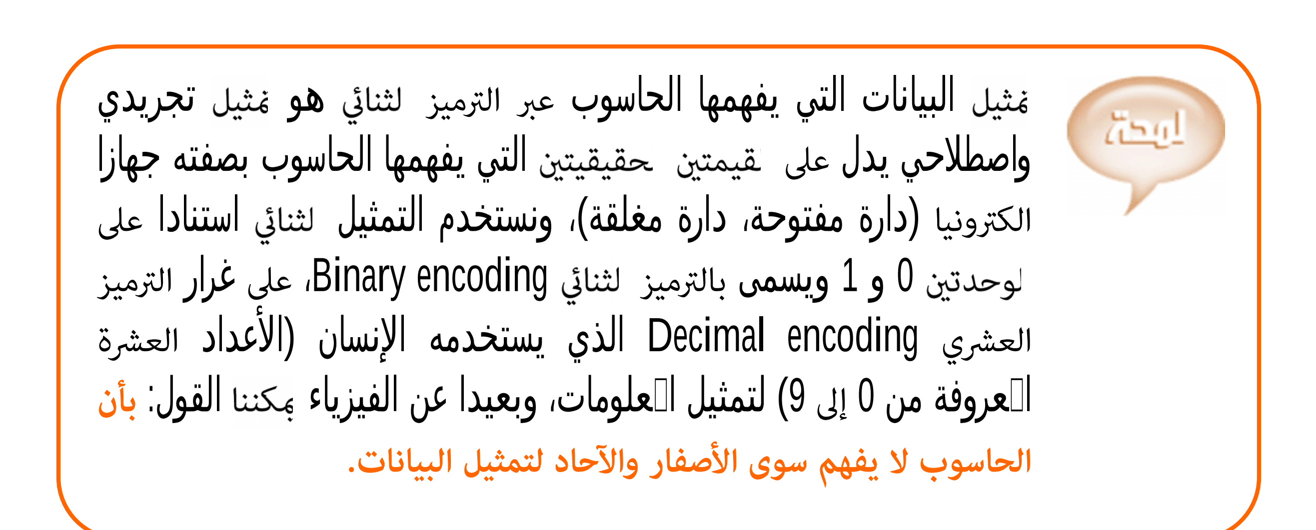
الأجهزة Devices:

وهي كل الأجهزة الموصولة بالحاسوب وهناك من يقسمها إلى أجهزة الإدخال Input devices : لوحة المفاتيح، سكانر، قارىء الأقراص،...إلخ. وأجهزة إخراج Output Devices: الشاشة، الطابعة، مكبرات الصوت،...إلخ. وأجهزة التخزين Storage Devices: أقراص صلبة، مفاتيح اليو أس بي، الأقراص، الديسكيت، ...إلخ.

اللغة التي يفهمها الحاسوب

رغم عظم المهام والعمليات التي يقوم بها الحاسوب، إلا أنه ليس ذو ذكاء خارق قياسا مع العقل البشري، فهو لا يفهم سوى رقمين 0 و 1 (وهذا تقدير اصطلاحي فقط لا علاقة له بما يتم فيزيائيا على مستوى الحاسوب)، كل البيانات سواء كانت عبارة عن فيديو، صورة، صوت، أو أي ملف آخر، فإن الحاسوب لا يراها سوى على شكل سلاسل من الأصفار والآحاد المخزنة في الذاكرة والتي تتم معالجتها وقراءتها بواسطة وحدة المعالجة، لكي تترجم إلى الشكل الذي نراها عليه.

أصفار وآحاد هي لغة الحاسوب، ولأنهما إثنان "2" سميت هذه اللغة ب "اللغة الثنائية Binary Language"، رغما أن هذه التسمية في المعلوميات الغرض الوحيد منها هو تبيان أن الحاسوب يفهم قيمتين متعارضتين فقط، وتم استخدام الأرقام 0 و 1 دلالة على ذلك، والأصل أن المسألة الكترونية، تنبني على التيار المتدفق، والمعلومات هي تسلسل لحالة التيار، لذلك تجد البعض يمثل اللغة الثنائية بالعبارة التالية "تيار يمر، تيار لا يمر" كناية على طبيعة البيانات التي يفهمها الحاسوب. هذا ما يقع فيزيائيا، لكن لفهم هذه المسائل تقنيا، يتم استخدام الثنائي 0 و 1 لتمثيل البيانات.



تمثيل البيانات التي يفهمها الحاسوب عبر الترميز الثنائي هو تمثيل تجريدي واصطلاحي يدل على قيمتين حقيقيتين التي يفهمها الحاسوب بصفته جهازا الكترونيا (دارة مفتوحة، دارة مغلقة)، ونستخدم التمثيل الثنائي استنادا على الوحدتين 0 و 1 ويسمى بالترميز الثنائي Binary encoding، على غرار الترميز العشري Decimal encoding الذي يستخدمه الإنسان (الأعداد العشرة المعروفة من 0 إلى 9) لتمثيل المعلومات، وبعيدا عن الفيزياء يمكننا القول: **بأن الحاسوب لا يفهم سوى الأصفار والآحاد لتمثيل البيانات.**

الترميز العشري

وهو الترميز المتعارف عليه عندنا نحن البشر، والذي يقوم على الأعداد العشرة المعروفة، التي تبدأ بصفر وتنتهي بتسعة (0، 1، 2، 3، 4، 5، 6، 7، 8، 9) ويتم الدمج بين هذه الأعداد للحصول على قيمة رقمية لها دلالتها، وهو أيضا ترميز اصطلاحي وتجريدي، اتفق عليه البشر لتمثيل الأشياء عدديا، فلو أتينا برجل من عصور ما قبل اختراع الأعداد، وقلنا له: 3، فلن يعي ما الذي نقصده بقولنا هذا، لكن إن قدمنا إليه ثلاث تفاحات، سيعرف أن عددها مختلف عن تفاحتين، فالترميز هو تجريدي فقط لتسهيل تمثيل الأشياء عدديا.

ويسمى هذا النوع من الترميز بالترميز العشري أو التمثيل العشري أو النظام العشري، لأنه يستخدم الرقم 10 لتبسيط وتفكيك الأعداد، فمثلا لو أخذنا الرقم التالي: 2897، فهو يتكون من الأجزاء التالية:

الوحدات	العشرات	المئات	الآلاف
7	9	8	2

الرقم 2 يمثل الجزء الخاص بالآلاف، وكما هو معلوم فالألف الواحد يضم عشر مئات، أي أن:

1000 = 10 * 100
2 * 1000 = 2 * 10 * 100
2000 = 2 * 1000

الرقم 8 يمثل الجزء الخاص بالمئات، والمئة الواحدة تحتوي على عشر عشرات، أي أن:

100 = 10 * 10

8 * 100 = 8 * 10 * 10

800 = 8 * 100

الرقم 9 يمثل الجزء الخاص بالعشرات، والعشرة الواحدة تضم عشر وحدات، أي أن:

10 = 10 * 1

9 * 10 = 9 * 10 * 1

90 = 9 * 10

الرقم 7 يمثل الوحدات، والوحدة هي أصغر جزء في الترميز العشري، ولا تحتوي إلا على نفسها، أي أن:

1 = 1 * 1

7 * 1 = 7 * 1 * 1

7 = 7 * 1

بالاعتماد على التقسيم الرياضي أعلاه، فإن العدد 2897 يساوي:

2897 = (2 * 1000) + (8 * 100) + (9 * 10) + (7 * 1)

الآن سنحوله إلى الترميز العشري بكل بساطة:

2897 = (2 * 10 * 10 * 10) + (8 * 10 * 10) + (9 * 10) + 7

وفي الرياضيات يتم تلخيص السطر أعلاه عبر استعمال القوى بالشكل التالي:

$2897 = (2 * 10^3) + (8 * 10^2) + (9 * 10^1) + (7 * 10^0)$

مع العلم أن:

10^3	10*10*10=1000
10^2	10*10=100
10^1	10
10^0	1

الترميز الثنائي

مفهوم الوحدة Bit

الوحدة Bit هي أصغر وحدة لقياس البيانات الممثلة في الترميز الثنائي المفهوم من طرف الحاسوب، وهي إما 0 أو 1.

1. وحدة واحدة تطرح أمامنا احتمالين وهما: صفر أو واحد (0 or 1)
2. وحدتان تطرحان أمامنا أربع احتمالات 2^2 وهم: صفر و صفر (0 and 0)، صفر وواحد (0 and 1)، واحد وصفر (1 and 0)، واحد وواحد (1 and 1)
3. أربع وحدات تطرح أمامنا احتمال بالعدد التالي: 2^4 (2*2*2*2) أي 16 احتمال:

0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1
0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1
1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1

ثماني وحدات تشكل ما يسمى بالبايت Byte، أي أن 1 Byte = 8 Bits، وهذا العدد من الوحدات يعطينا 28 احتمال، أي 256 احتمال ممكن.

ويمكن أن نلخص ماسبق في العلاقة التالية:

N وحدة (bit) ⟸ 2N احتمال ممكن.

مفهوم البايت Byte

البايت كما رأينا قبل قليل، هو وحدة لقياس البيانات وهو يتكون من ثماني وحدات، لذلك قد تسمع بالكيلو بايت في قياس حجم الملفات والميغا بايت، والجيغا بايت، وباستخدام العلاقة التي كتبناها أعلاه فإنك تستطيع التحويل بين الوحدات بكل سهولة بالاعتماد على الجدول التالي:

1 KiloByte (KB) = 210 Byte = 1024 Byte

1 MegaByte (MB) = 210 KiloByte = 1024 KiloByte

1 GegaByte (GB) = 210 MegaByte = 1024 MegaByte

1 TeraByte (GB) = 210 GegaByte = 1024 GegaByte

تحويل البيانات الثنائية إلى بيانات عشرية

إذن فالترميز الثنائي يقوم على القاعدة 2، ويتم تمثيله دائما بالاعتماد على القوى المتعلقة بالرقم 2، فلو افترضنا أن عندنا البايت التالي: 10110010

إذا أردنا كتابة هذا البايت بالترميز العشري فإنه علينا النظر إلى عدد الوحدات المكونة للبايت ونقص منه واحدا، ونجعله أسا للأساس 2 ونضربه في أول وحدة ابتداء من الشمال، وكلما انتقلنا إلى وحدة أخرى نقصنا قيمة الأس بواحد، أي أن البايت أعلاه يتم تحويله إلى الترميز العشري بالطريقة الآتية:

$$10110010 = (1 * 2^7)+(0 * 2^6)+(1 * 2^5)+(1 * 2^4)+(0 * 2^3)+(0 * 2^2)+(1 * 2^1)+(0 * 2^0)$$

أي أن:

$$10110010 = (1 * 128)+(0 * 64)+(1 * 32)+(1 * 16)+(0 * 8)+(0 * 4)+(1 * 2)+(0 * 1)$$

أي أن:

$$10110010 = (128)+(0)+(32)+(16)+(0)+(0)+(2)+(0)$$

أي أن:

$$10110010 = 178$$

لتوضيح ترميز كل قيمة نكتبها بالشكل التالي:

$$(10110010)_8 = (178)_{10}$$

ولتدعيم الشرح بمثل آخر، سنأخذ المتتالية الثنائية التالية المتكونة من ست وحدات فقط 101010، وسنقوم بتحويلها إلى الترميز العشري بنفس الطريقة السابقة:

$$(101010)_2 = (1 * 2^5)+(0 * 2^4)+(1 * 2^3)+(0 * 2^2)+(1 * 2^1)+(0 * 2^0)$$

$$(101010)_2 = (32)+(0)+(8)+(0)+(2)+(0)$$

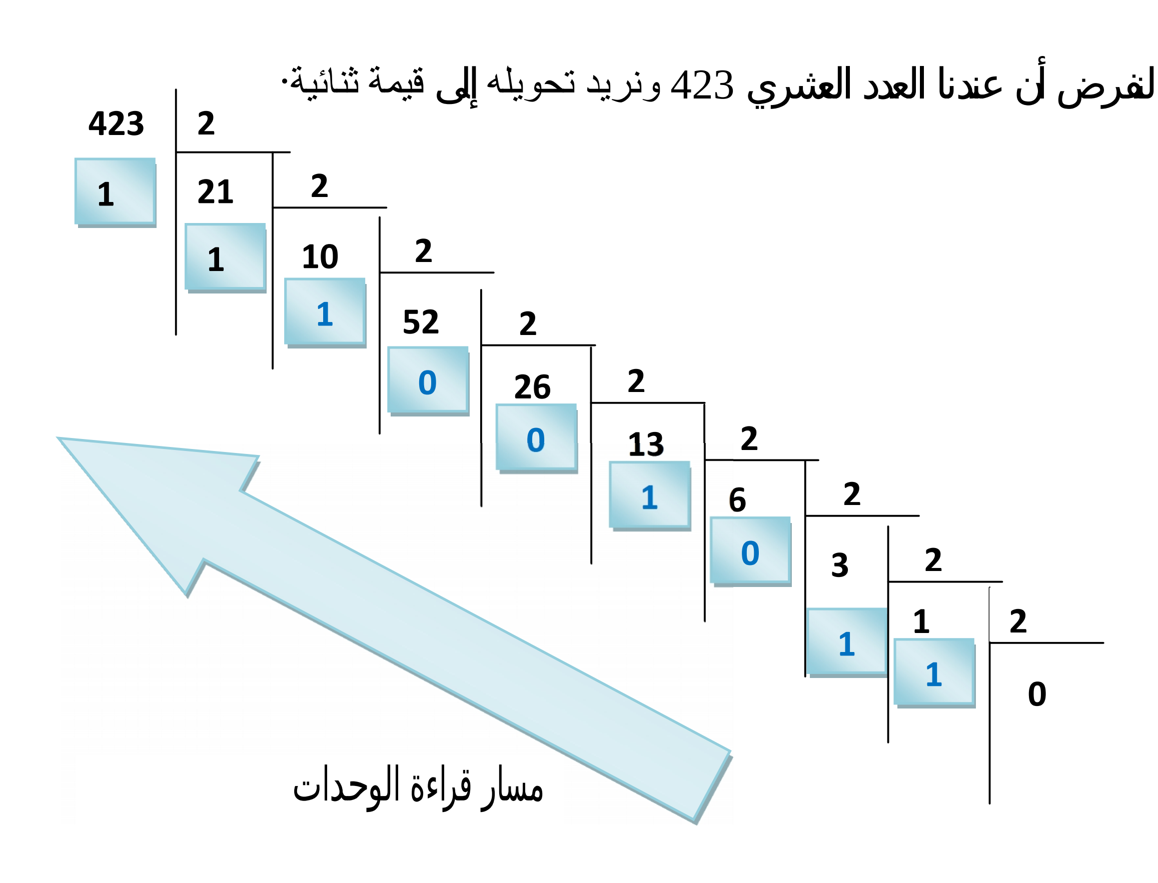
$$(101010)_2 = (42)_{10}$$

تحويل البيانات العشرية إلى بيانات ثنائية

الطريقة الأولى:

لتحويل البيانات العشرية إلى بيانات ثنائية توجد طرق كثيرة، أبرزها عملية قسمة العدد في الترميز العشري على العدد 2 بالطريقة التالية.

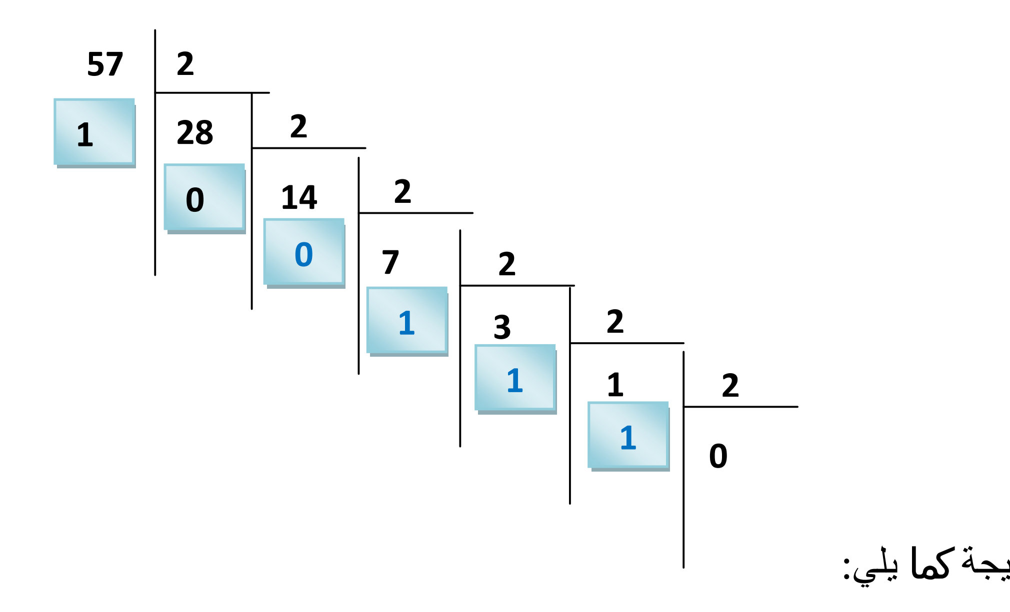
لنفرض أن عندنا العدد العشري 423 ونريد تحويله إلى قيمة ثنائية.



بعد أن ننتهي من عملية القسمة ويكون الحاصل هو صفر، نقوم بتجميع الوحدات المحصول عليها ونقرأها من الأخير كما يعرض السهم أعلاه لكي نحصل على القيمة الثنائية المرادفة للعدد العشري، وفي المثل السابق حصلنا على النتيجة التالية:

$$(423)_{10} = (110100111)_{2}$$

وهذا مثال آخر لكي نفهم كيفية تحويل العدد في الترميز العشري إلى عدد في الترميز الثنائي عبر عملية القسمة:



أي أن النتيجة كما يلي:

$(57)_{10} = (111001)_2$

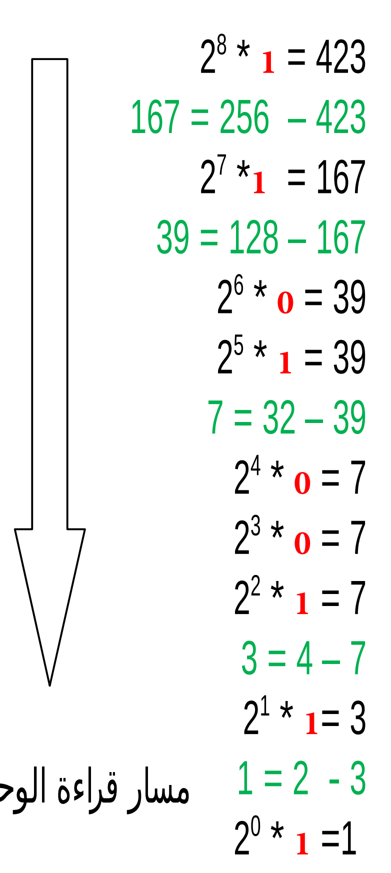
الطريقة الثانية:

أما إن كانت عندك دراية بالحساب، فهنالك طريقة أخرى أسهل من القسمة، وتكون عبر أخذ أكبر قيمة من العدد في الترميز العشري شريطة أن يكون أسها هو 2، ونطرحها من العدد، ونأخذ الباقي ونطبق عليه نفس الأمر، قد تحتاج إلى الجدول التالي لمساعدتك في حساب قيمة كل أس:

2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32

2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1024

لو أخذنا نفس المثال الأول الذي استعرضنا فيه كيفية تحويل العدد 423 إلى الترميز الثنائي فإننا سنطبق عليه ما يلي:



بعد تجميع الوحدات وفق مسار القراءة الذي يدل عليه السهم، سنحصل على نفس النتيجة السابقة وهي:

$(423)_{10} = (110100111)_2$

الآن سنجرب نفس الطريقة على العدد 26 لنحصل على مقابله في الترميز الثنائي:

$2^4 * 1 = 26$

$10 = 16 - 26$

$2^3 * 1 = 10$

$2 = 8 – 10$

$2^2 * 0 = 2$

$2^1 * 1 = 2$

$0 = 2 – 2$

$2^0 * 0 = 0$

إذا جمعنا الوحدات فسوف نحصل على ما يلي:

$(26)_{10} = (11010)_2$

الترميز الثماني

قلنا فيماسبق أن الحاسوب يقوم بمعالجة البيانات على شكل وحدات Bits، علما أن الوحدة لا تخزن لوحدها في الذاكرة وإنما تخزن على شكل مجموعة مؤلفة من ثماني وحدات يصطلح عليها بالبايت Byte، أي أن البايت الواحد يساوي ثماني وحدات 1 Byte=8 Bits.

ويسمح لنا البايت الواحد بتخزين حرف أو رقم، ويصطلح على السلسلة المكونة من 16 وحدة (أي 2 بايت) ب"الكلمة" وفي اللغة الانجليزية Word، وعلى هذا المنوال تأتي الكلمة DWord التي تستعمل للدلالة على سلسلة مكونة من كلمتين Double Word أي أربع بايتات 4 Bytes أي 32 وحدة 32 Bits.

الترميز الثماني هو ترميز يقوم على تمثيل البيانات عبر أعداد قاعدتها ثمانية، ويسمح لنا باستخدام ثمانية أرقام فقط وهي 0، 1، 2، 3، 4، 5، 6، 7. ويتم استخدامه من باب زيادة حجم البيانات المعالجة، ولتسهيل قراءة البيانات الثنائية.

تحويل البيانات العشرية إلى بيانات ثمانية والعكس:

لتحويل البيانات من التمثيل العشري إلى بيانات ممثلة ثمانيا، فالطريقة كما قلنا قبل قليل، نحتاج فقط إلى تفكيك الأعداد للحصول على أعداد أساسها الرقم 8، وهذا مثال يوضح كيفية عمل ذلك:

$(15)_{10} = 8 + 7$

$(15)_{10} = (1 * 8^1)+(7 * 8^0)$

$\mathbf{(15)_{10} = (17)_8}$

لدعم الفهم ركز في المثل الثاني:

$(153)_{10} = (128) + (24) + (1)$

$(153)_{10} = (2 * 8^2) + (3 * 8^1) + (1 * 8^0)$

$\mathbf{(153)_{10} = (231)_8}$

ويمكننا تحويل البيانات في الاتجاه العكسي، أي من التمثيل الثماني إلى التمثيل العشري عبر طريقة سهلة جدا تقوم على تفكيك القيمة إلى أعداد قاعدتها ثمانية، فلو أخذنا القيمة الثمانية التالية $(340)_8$ فإن تحويلها إلى الترميز العشري يكون بالشكل التالي:

$(340)_8 = (3 * 8^2) + (4 * 8^1) + (0 * 8^0)$

$(340)_8 = (3 * 64) + (4 * 8) + (0 * 1)$

$(340)_8 = (192) + (32) + (0)$

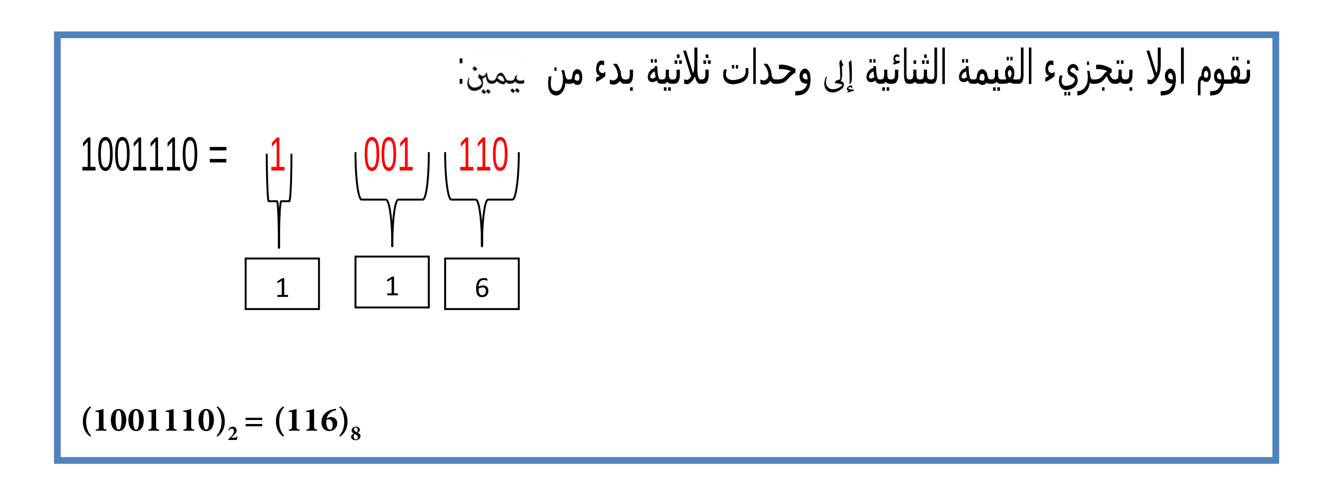
$\mathbf{(340)_8 = (224)_{10}}$

تحويل البيانات الثنائية إلى بيانات ثمانية والعكس:

تعتبر عملية تحويل البيانات من صيغتها الثنائية إلى الصيغة الثمانية من أسهل العمليات، بيد أن علينا تقسيم القيمة الثنائية إلى أجزاء مكونة من ثلاث وحدات بدء من اليمين، ونقوم بتأويل كل جزء ثلاثي في الصيغة الثنائية بمرادفه الثماني عبر استخدام الجدول التالي:

القيمة الثنائية	القيمة الثمانية
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

أي لو أننا أخذنا السلسلة الثنائية التالية: 1001110 فإن عملية تحويلها إلى التمثيل الثماني يكون بالشكل التالي:



وهذا مثل آخر لدعم الفهم:

$(100111001110)_2$ =	100	111	001	110
$(100111001110)_2$ =	4	7	1	6
$(100111001110)_2 = (4716)_8$				

أما تحويل البيانات الثمانية إلى بيانات ثنائية فيكون عبر إعطاء القيمة الثنائية لكل جزء من أجزاء القيمة الثمانية:

$(234)_8$ = 010	011	100
$(234)_8 = (10\ 011\ 100)_2$		

الترميز الست عشري

يعتبر الترميز الست عشري من أبرز أنواع الترميز المستخدمة في المعلوميات لأنه يسمح بقراءة السلاسل الثنائية الطويلة بشكل مختصر، عبر اعتماد التمثيل الرقمي المستند على 16 حرفا وهي الأرقام المعروفة في النظام العشري (0، 1، 2، 3، 4، 5، 6، 7، 8، 9) بالإضافة إلى الحروف الستة الأولى من الحروف الهجائية اللاتينية (a، b، c، d، e، f)التي يستعاض بها عن الأرقام التالية:11، 12، 13، 14، 15، 16.

هذا التمثيل الذي يمكننا من استخدام 16 حرفا، يجعل البيانات سهلة القراءة لأنها تكتب بشكل مختصر، ولك أن تقارن بين القيم التالية وأيها أشد استيعابا عند العقل البشري:

$(10011100)_2 = (9C)_{16}$

$(11110100011001011)_2 = (1E8CB)_{16}$

$(1100000101011111001110110)_2 = (182BE76)_{16}$

من غير شك فالتمثيل الست عشري أسهل قراءة وأكثر استيعابا من التمثيل الثنائي، لأنه يعرض المعلومة عبر حروف قليلة ومحدودة في الوقت الذي تكتب فيه هذه المعلومة عبر سلاسل طويلة من الأصفار والآحاد التي من الصعب قراءتها ترتيبيا فضلا عن تذكرها أو معالجتها من قبل الدماغ البشري.

جدول الأعداد الست عشرية

الرقم الست عشري	الرقم العشري
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

تحويل البيانات من الترميز العشري إلى الترميز الست عشري

تقريبا نفس الكلام سيتكرر، فكل طرق التحويل التي رأيناها متماثلة في عملية التحويل مع الأخذ بعين الاعتبار القاعدة التي يستند إليها كل نظام، فالترميز الثنائي يستند إلى القاعدة 2، والترميز الثماني يستند إلى القاعدة 8، والترميز العشري يستند إلى القاعدة العشرية، وبالمثل يستند النظام الست عشري إلى القاعدة 16، وهذا مثال يبين لنا كيف نقوم بتحويل بيانات ممثلة في النظام العشري إلى بيانات ممثلة في النظام الست عشري:

$23 = (1 * 16^1) + (7 * 16^0)$

$(23)_{10} = (17)_{16}$

___

$145 = (9 * 16^1) + (1 * 16^0)$

$(145)_{10} = (91)_{16}$

___

$367 = (1 * 16^2) + (6 * 16^1) + (15 * 16^0)$

$367 = (1 * 16^2) + (6 * 16^1) + (15 * 16^0)$

$367 = (1 * 16^2) + (6 * 16^1) + (F * 16^0)$

$(367)_{10} = (16F)_{16}$

بما ان الرقم 15 في النظام الست عشري هو الحرف F (راجع جدول الأعداد الست عشرية)

تحويل البيانات من الترميز الست عشري إلى الترميز العشري

الطريقة بسيطة جدا وتكون عبر تفكيك العدد الست عشري إلى أعداد مفصلة، ثم نعطي كل عدد في الترميز الست عشري قيمته في الترميز العشري بالاستناد إلى جدول الأعداد الست عشرية الذي أوردناه في الأعلى، وطريقة التحويل تتم بالشكل التالي:

$(54)_{16} = (5 * 16^1) + (4 * 16^0)$

$(54)_{16} = (80) + (4)$

$\mathbf{(54)_{16} = (84)_{10}}$

$(89)_{16} = (8 * 16^1) + (9 * 16^0)$

$(89)_{16} = (128) + (9)$

$\mathbf{(89)_{16} = (137)_{10}}$

$(F3D)_{16} = (15 * 16^2)+(3 * 16^1) + (13 * 16^0)$

$(F3D)_{16} = (3840) + (48) + (13)$

$\mathbf{(F3D)_{16} = (3901)_{10}}$

تحويل البيانات من الترميز الثنائي إلى الترميز الست عشري

تجدر الإشارة إلى أن عملية التحويل من النظام الثنائي إلى النظام الست عشري تتم عبر تجميع كل أربع وحدات على حدة، ثم تستبدل هذه الرباعية بمقابلها في النظام الست عشري وفق الجدول التالي:

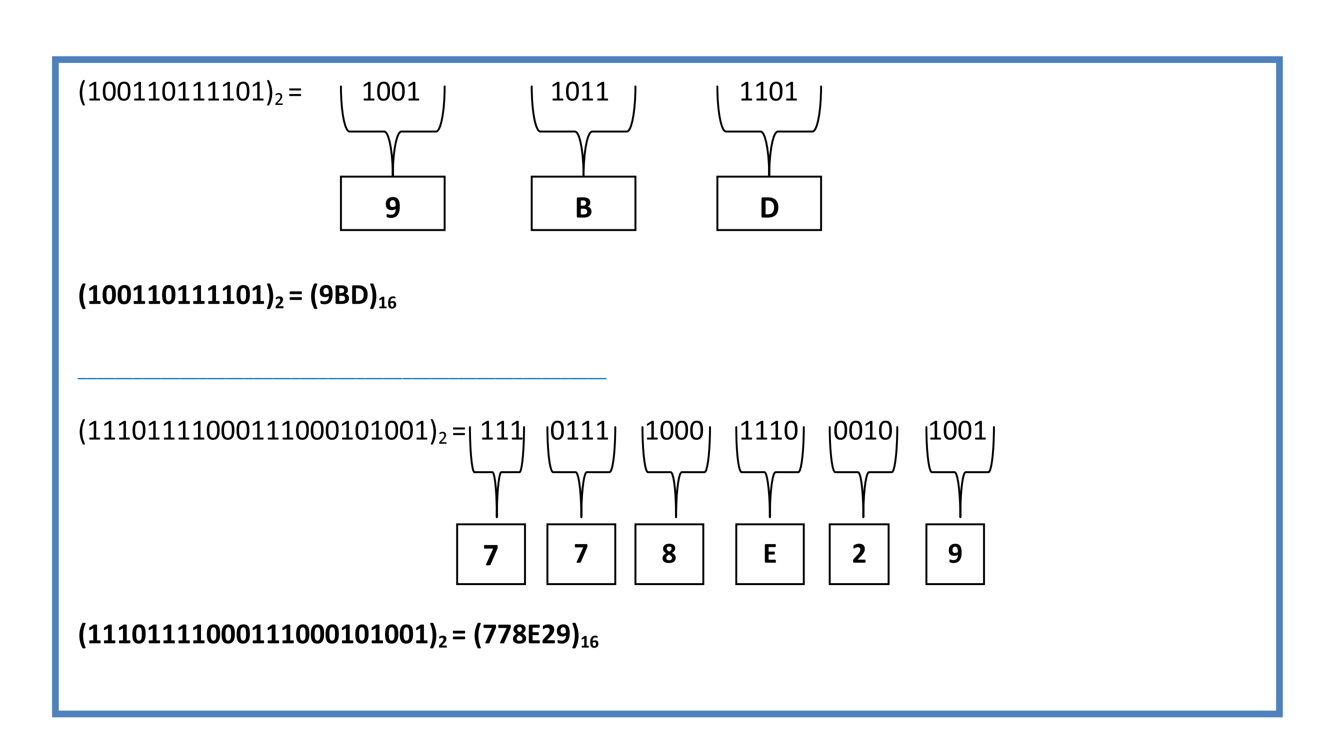
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101

8	1000
9	1001
A	1010
B	1011
C	1100
D	1101

6	0110
7	0111

E	1110
F	1111

وهذه الأمثلة توضح الكلام أعلاه:



تحويل البيانات من الترميز الست عشري إلى الترميز الثنائي

أما طريقة تحويل البيانات الست عشرية إلى البيانات الثنائية فيكون عبر قلب الطريقة السابقة، بحيث نقوم بتفكيك العدد الست عشري إلى أجزاء ونأخذ كل جزء على حدة ونستبدله بمقابله في النظام الثنائي بالاعتماد على الجدول السابق، وهذه الأمثلة تبين لنا كيفية القيام بهذا النوع من التحويل:

$(3D)_{16} = (0011 \quad 1101)_2$

لان الرقم 3 في الترميز لثنائي هو 0011، والرقم D هو 1101 (راجع الجدول السابق)

$(6FE89A)_{16} = (0110 \quad 1111 \quad 1110 \quad 1000 \quad 1001 \quad 1010)_2$

$(458ECB)_{16} = (10001011000111011001011)_2$

$(12CFE8B74B)_{16} = (1001011001111111010001011011101001011)_2$

تحويل البيانات من الترميز الثماني إلى الترميز الست عشري

يمكنك تحويل القيمة الثمانية إلى النظام الست عشري عبر تحويلها أولا إلى النظام الثنائي ثم تحويل الناتج إلى النظام الست عشري بنفس الطريقة التي رأيناها في الفقرة السابقة:

$(12)_8 = (001 \quad 110)_2$

$(12)_8 = (1110)_2$

$(1110)_2 = (E)_{16}$

$\mathbf{(12)_8 = (E)_{16}}$

لان الرقم 1 في النظام ثماني يساوي 001 في النظام لثنائي، والرقم 2 يساوي 110 في النظام لثنائي (راجع جدول التحويل من النظام ثماني إلى النظام الثنائي الموجود في الصفحة 20)

$(345)_8 = (011 \quad 100 \quad 101)_2$

$(345)_8 = (11100101)_2$

$(11100101)_2 = (E5)_{16}$

$\mathbf{(345)_8 = (E5)_{16}}$

$(2431)_8 = (010 \quad 100 \quad 011 \quad 001)_2$

$(2431)_8 = (10100011001)_2$

$(10100011001)_2 = (519)_{16}$

$\mathbf{(2431)_8 = (519)_{16}}$

تحويل البيانات من الترميز الست عشري إلى الترميز الثماني

نستطيع تحويل البيانات الممثلة في النظام الست عشري إلى بيانات أخرى ممثلة في النظام الثماني من خلال استخدام التحويل الثنائي كوسيط بين النظامين، بنفس الطريقة السابقة، لكن

هذه المرة عبر تحويل القيمة الست عشرية إلى قيمة ثنائية، ثم تحويل الناتج إلى قيمة ثمانية، كما تعرض لنا الأمثلة التالية:

$(6EB)_{16} = (0110 \quad 1110 \quad 1011)_2$

راجع طريقة تحويل البيانات الست عشرية إلى بيانات ثنائية في الصفحة 25

$(6EB)_{16} = (11011101011)_2$

$(11011101011)_2 = (3353)_8$

$\mathbf{(6EB)_{16} = (3353)_8}$

$(EE56FD)_{16} = (1110 \quad 1110 \quad 0101 \quad 0110 \quad 1111 \quad 1101)_2$

$(EE56FD)_{16} = (111011100101011011111101)_2$

$(111011100101011011111101)_2 = (73453375)_8$

$\mathbf{(EE56FD)_{16} = (73453375)_8}$

سلسلة تمارين حول أنظمة تمثيل البيانات

التمرين 1:

1. اعط تعريفا موجزا لجهاز الحاسوب.
2. ما هما الجزءان المكونان للحاسوب، وقم بتعريف كل منهما؟
3. ماهي أهم المكونات المادية لجهاز الحاسوب؟
4. ماهو دور الذاكرة الرئيسية؟
5. ماهو دور وحدة معالجة البيانات؟
6. ماهي أنواع الأجهزة الممكن وصلها مع الحاسوب؟
7. اذكر المراحل التي يمر منها تنفيذ أي برنامج على جهاز الحاسوب، مع تحديد المكونات المتدخلة في العملية موضحا دور كل مكون مادي في عملية تنفيذ البرامج.
8. ماهو نظام ترميز البيانات الذي يفهمه الحاسوب؟
9. ماهي أشهر أنواع أنظمة تمثيل البيانات؟ واذكر القاعدة الرقمية لكل نظام.
10. لخص في ورقة طريقة الانتقال من نظام إلى نظام آخر، ليسهل عليك الإجابة على التمارين التالية.

التمرين 2:

قم بحساب ناتج التحويل بين وحدات قياس البيانات التالية:

- 1 جيغا بايت كم يساوي من كيلو بايت؟
- 75 ميغا بايت كم يساوي من كيلو بايت؟
- 12 كيلو بايت كم يساوي من بايت؟
- 1750 بايت كم يساوي من كيلوبايت؟
- 5680 كيلوبايت كم يساوي من ميغا بيت؟

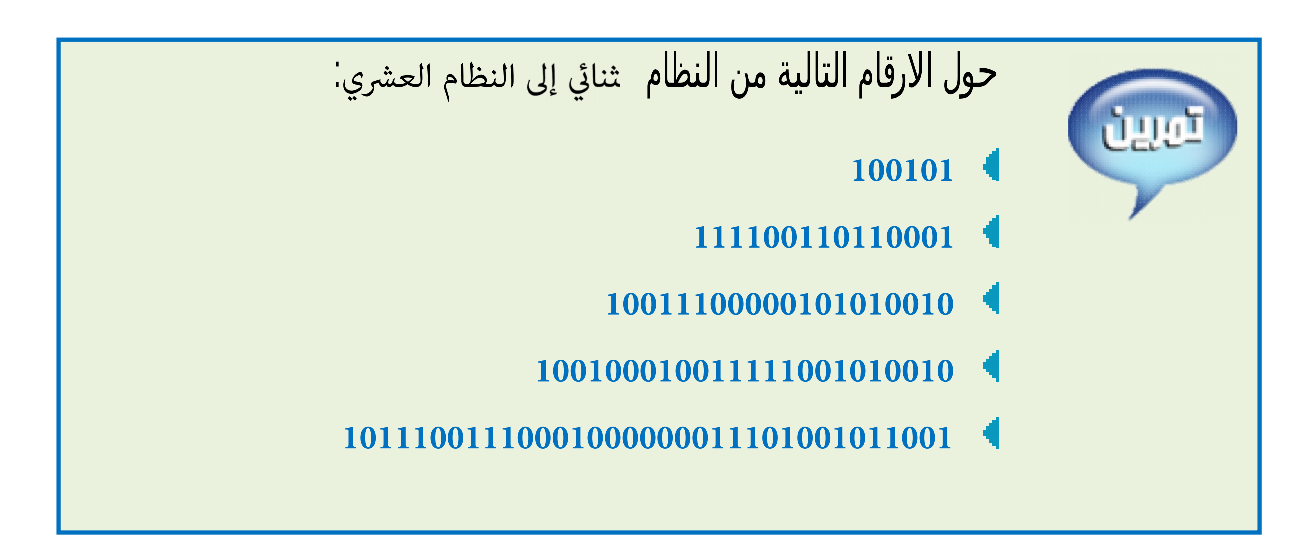
التمرين 3:

تمرين

حول الارقام التالية من النظام العشري إلى النظام لثنائي:

- 7
- 162
- 943
- 5872
- 761452

التمرين 4:



تمرين

حول الأرقام التالية من النظام ثنائي إلى النظام العشري:

- 100101
- 111100110110001
- 1001110000010101010010
- 1001000100111111001010010
- 10111001110001000000001110100101 1001

التمرين 5:



تمرين

حول الأرقام التالية من النظام ثماني إلى النظام لثنائي:

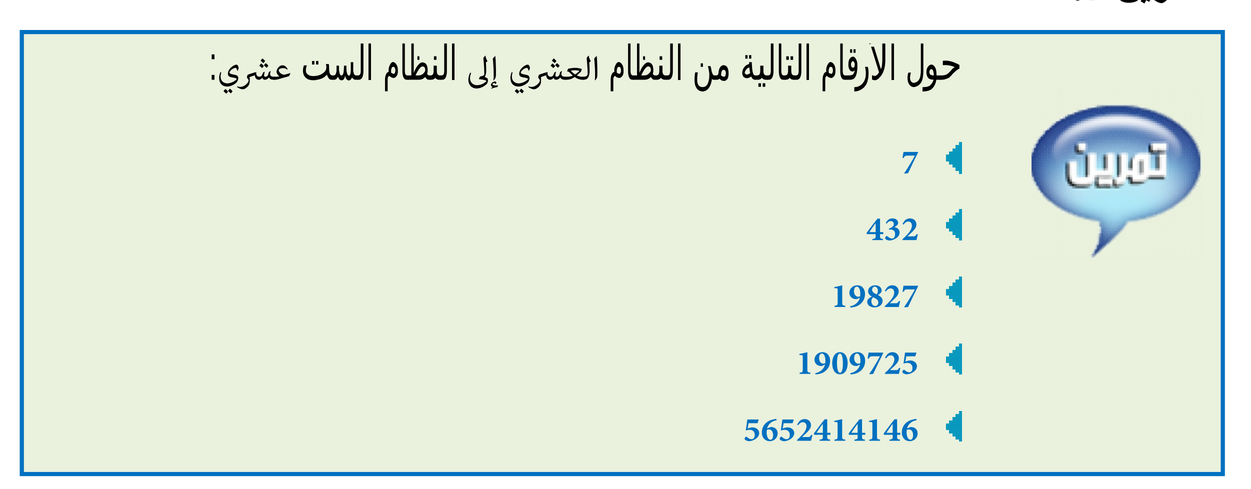
- 56
- 123
- 6761
- 671512
- 13726552

التمرين 6:

حول الأرقام التالية من النظام ثنائي إلى ظام الثماني:

- 10101
- 1110111
- 1001000110
- 11111011101
- 1010101010101111010

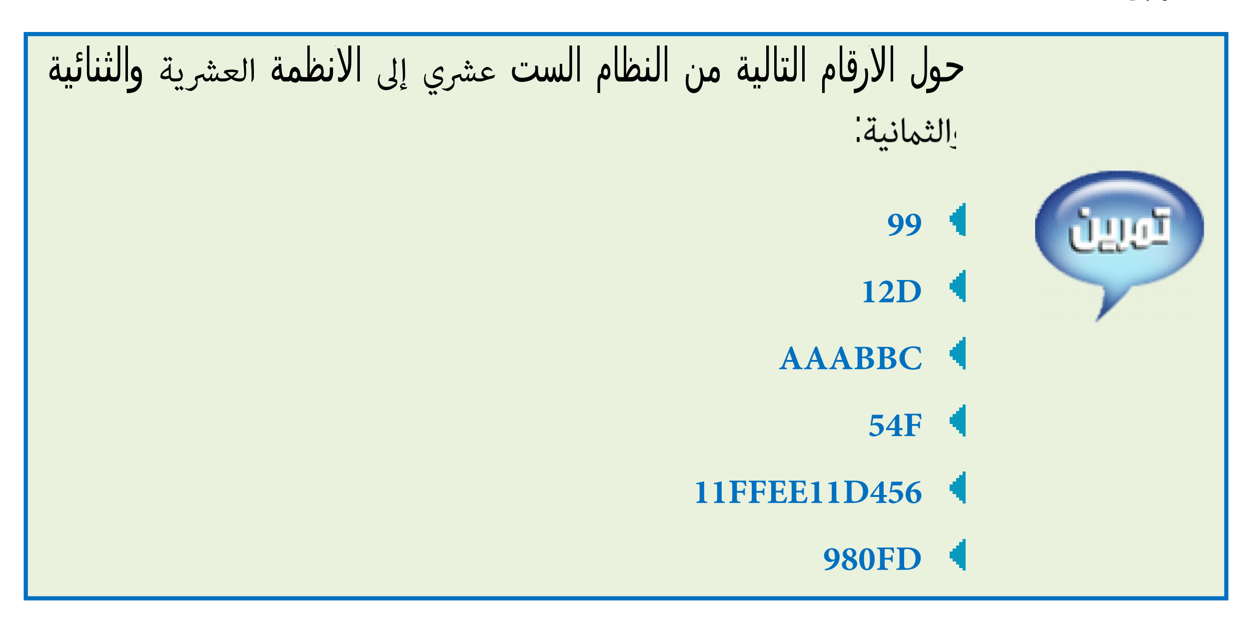
التمرين 7:



حول الأرقام التالية من النظام العشري إلى النظام الست عشري:

- 7
- 432
- 19827
- 1909725
- 5652414146

التمرين 8:



حول الأرقام التالية من النظام الست عشري إلى الانظمة العشرية والثنائية والثمانية:

- 99
- 12D
- AAABBC
- 54F
- 11FFEE11D456
- 980FD

الفصل الثاني: العمليات الحسابية في النظام الثنائي

العمليات الحسابية في النظام الثنائي

نستطيع القيام بالعمليات الرياضية المألوفة من قبيل الجمع والطرح والضرب والقسمة على الأعداد الثنائية، بنفس الطريقة العادية التي نستخدمها في نظامنا العشري، لكن قد يتساءل البعض لماذا سنحتاج إلى حساب الأعداد الثنائية، وهل علي أن أقوم بهذه العمليات كلما أردت أن أبرمج شيئا في المستقبل، أبشرك بأنك لست ملزما بإتقان هذه العمليات ولا حتى عمليات التحويل التي رأيناها في الفصل السابق، لأن ذلك كله من اختصاصات الأجهزة المشغلة للبرامج، فالحاسوب هو الموكل بالقيام بكل هذه العمليات وليس المبرمج لأن اللغة الثنائية كما ذكرنا في بداية الكتاب هي اللغة الخاصة بالحاسوب وبالتالي كل ما يترتب عنها هو مرتبط بالحاسوب، لكن من باب فهم حقيقة ما يجري على مستوى الحاسوب عند معالجته للبيانات فإننا نتعلم هذه الأمور لكي نفهم كيف تفكر هذه الآلة التي نخاطبها، وأعتقد و الحمد أن الرؤية وضحت أكثر وصرنا على دراية بالعمليات التي تجري على مستوى الحاسوب لتنفيذ الأوامر الموجهة إليه، وهذا هو مرمانا ومسعانا.

طيب تعالوا بنا الآن نطرح السؤال التالي: حينما نوجه إلى الحاسوب أمر إضافة عددين عشريين مالذي يحدث؟

أولا يتم تحويل هذين العددين العشريين إلى عددين ثنائيين، ثم يتم الجمع بينهما، وتحويل النتيجة إلى النظام العشري من أجل طباعتها للمستخدم.

عملية الجمع:

في النظام العشري الذي نستخدمه نحن البشر يكون مجموع واحد زائد واحد هو اثنان 2=1+1

في النظام الثنائي نتيجة واحد زائد واحد هي عشرة وليست إثنان، أي أن 1+1=10، وباقي الحسابات تتم بالشكل العادي، ويمكننا تلخيص حساب الجمع في النظام الثنائي في الجدول التالي:

+	0	1
0	0	1
1	1	10

بناء على الجدول أعلاه فإن عملية الجمع للأعداد الثنائية التالية تتم بهذا الشكل:

```
  10101            1101110           1001011
+  1001          +   10010          +1110010
 ______          _________          ________
  11110           10000000          10111101
```

```
  11000101110101
 +10100100101001
 _______________
 101101010011110
```

عملية الطرح:

في النظام الثنائي، تتم عملية الطرح بنفس الطريقة المألوفة والمستخدمة في النظام العشري، بحيث يتم طرح الرقم العلوي من الرقم السفلي، وإذا كان الرقم السفلي أكبر من الرقم العلوي يتم إضافة عشرة إلى هذا الأخير مع نقصان واحد من الرقم الموالي كما جرت العادة مع عملية الطرح، ولنفترض أننا في النظام العشري نود طرح 127 من الرقم 352، العملية ستكون بالشكل التالي:

```
   ¹
  352
 -1¹27
 -----
  225
```

في النظام الثنائي، إذا أردنا طرح 0 من واحد فإن النتيجة هي 1، وإذا أردنا طرح 1 من 1 فإن النتيجة هي صفر، وإذا أردنا طرح 0 من 0 فإن النتيجة هي صفر، أما إذا أردنا أن نطرح 1 من 0 فإننا هنا نحتاج إلى إضافة 1 إلى 0 ليصبح لدينا وحدتان مقابل وحدة واحدة وبالتالي تكون نتيجة طرح 1 من 0 هي 1.

وهذا مثال لتوضيح كيفية إنجاز عملية الطرح في النظام الثنائي:

```
    ¹
  1010
 -100¹1
 ------
  0001
```

في عملية الطرح أعلاه، قمنا بطرح القيمة 1001 من القيمة 1010، فأول ما كان لدينا هو طرح واحد من صفر، وبالعودة إلى القاعدة السابقة فإننا سنحتاج إلى إضافة واحد إلى صفر، فيصبح لدينا (1+0 أي صار لدينا وحدتان : 2 Bits) وبطرح وحدة واحدة من وحدتين فإن الناتج سيكون هو وحدة واحدة لذلك حصلنا على 1، ولتعميق الشرح تذكر أننا في النظام العشري إذا أردنا طرح رقم صغير من رقم كبير نضيف 1 وهو يمثل 10، فكذلك في النظام الثنائي نضيف 1 وهو يمثل 2.

ثم بعد ذلك نقصنا 1 من 1 (لأن الصفر السفلي انضاف إليه 1 الذي استلفناه في أول وحدة) وبالتالي حصلنا على صفر، بعد ذلك طرحنا صفر من صفر فحصلنا على صفر، ثم طرحنا واحد من واحد فحصلنا على صفر.

فيما يلي مجموعة من الأمثلة لفهم عملية الطرح جيدا:

101011 − 100101 = 000110

1001101 − 100101 = 0101000

1011010 − 101001 = 0110001

عملية الضرب:

عملية الضرب في النظام اثنائي هي نفسها في النظام العشري، نقوم بضرب كل رقم من العدد السفلي في جميع أرقام العدد العلوي، ثم نقوم بجمع نتائج كل جداء لنحصل على النتيجة النهائية، فيما يلي مثال على عملية الضرب في النظام العشري:

```
  123
×  64
-----
  492
+738
-----
 7872
```

وهذا الجدول يلخص نتيجة الضرب في النظام الثنائي:

X	0	1
0	0	0
1	0	1

وهذه أمثلة لمجموعة عمليات الضرب في النظام الثنائي:

```
  10
×  1
----
  10
```

```
  101
×  11
-----
  101
+101
-----
 1111
```

```
    1010
×    101
--------
    1010
+  0000
  10 10
--------
 11 0010
```

الفصل الثالث:

الخوارزميات

البرمجية

أصول وأبجديات البرمجة

ملاحظات مهمة قبل البدء

قبل أن نبدأ دراسة الخوارزمية أحب أن أنبه إلى مجموعة من الأمور المهمة:

- الطريقة التي تكتب بها الخوارزمية ليست ثابتة وقد تتعدد صيغة كتابتها لكن المهم والأساسي أن تكون الخطوات منطقية ومترابطة وقابلة للتحويل إلى لغة برمجية.
- الطريقة التي سنستعملها في هذا الكتاب طريقة شخصية تخلط بين الصيغة الكلاسيكية لكتابة الخوارزميات وبين البنية الرئيسية المشتركة بين لغات البرمجة المشهورة (جافا،سي شارب،سي،سي بلس بلس، بي إتش بي، ...) لكي لا يجد المتعلم صعوبة حينما يريد تطبيق ما فهمه من الكتاب على لغة برمجية معينة.
- الأوامر المكتوبة في الخوارزميات تسمى كود زائف Pseudo Code، أي أنها مجرد كتابة منطقية وليست كود حقيقي، ومعظم الأكواد الزائفة الواردة في الأمثلة ستجدها مرفقة بعناوين خضراء تسمى في البرمجة تعاليقا Comments وهي كلمات لا يتم تنفيذها ويكون دورها هو تنظيم الكود وعنونته ليكون سهل القراءة ومستساغ الفهم، وهذا مثال على التعاليق المستخدمة في الكتاب:

```
//هذا السطر الأخضر تعليق على الكود

/*
وهذا أيضا تعليق
لكنه موزع على أكثر من سطر
*/
```

ماهي الخوارزميات؟

الخوارزميات هي طريقة منطقية في التفكير نستخدمها من أجل الوصول إلى حل وضعية معينة عبر عدد محدد من الخطوات المنطقية المترابطة فيما بينها، وسميت بهذا الإسم نسبة إلى صاحبها أبي جعفر الخوارزمي رحمه ا ، الذي عاصر فترة الحكم العباسي وتحديدا عصر الخليفة المأمون، وكان الخوارزمي رحمه ا نابغا في علوم الحساب والفلك والجغرافيا وهو مؤسس علم الجبر ومؤسس الخوارزميات، وله العديد من الإنجازات العلمية العظيمة.

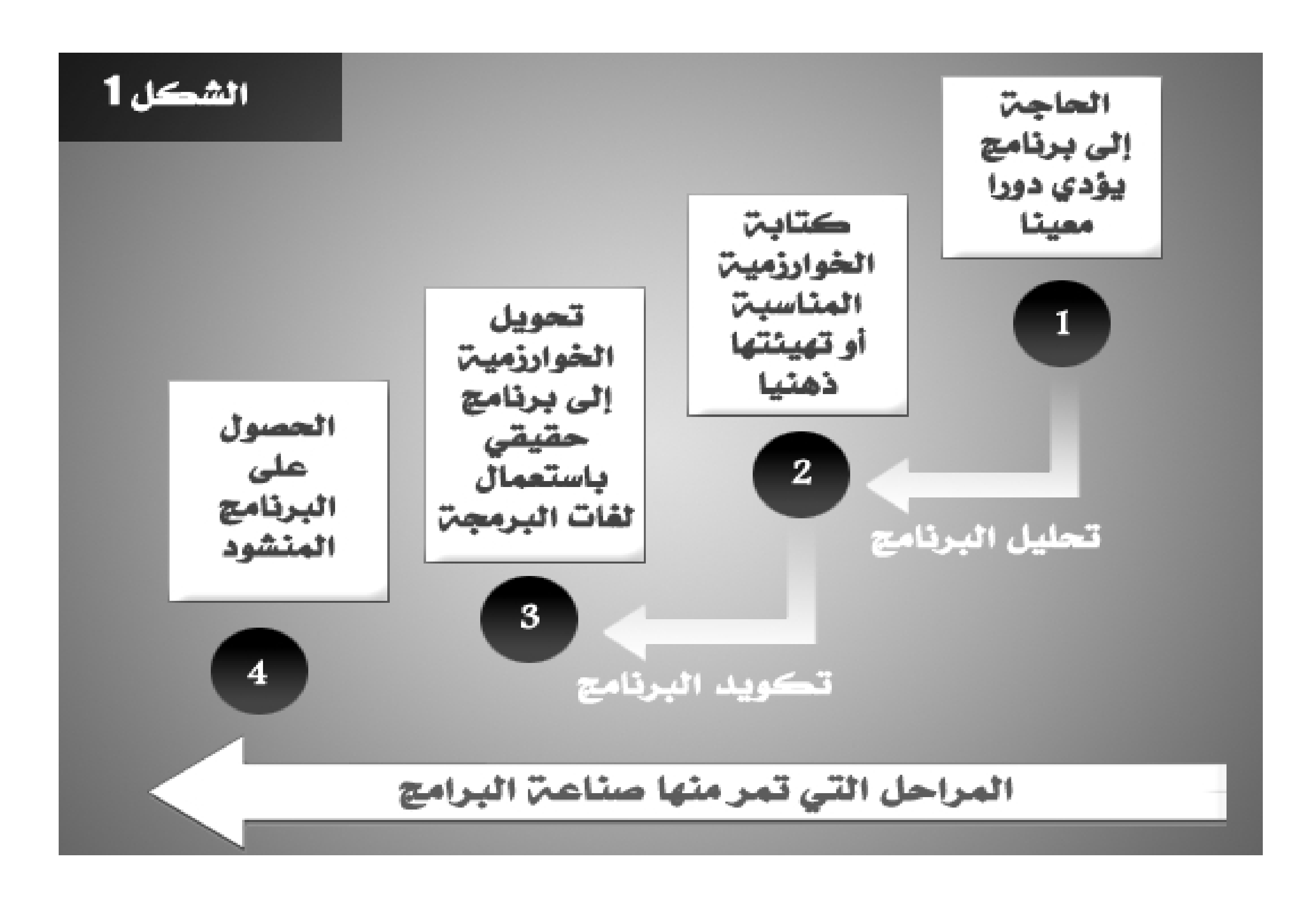
لو أردنا تبسيط الخوارزميات، فهي خطوات نقوم بها لحل مشكل معين، مثلا عندنا برنامج معين لكي نشغله على حاسوبنا فعلينا تنصيبه، أي اتباع مجموعة من الخطوات التي تؤدي لنا في الأخير تثبيت البرنامج وجعله قابلا للاشتغال على الحاسوب.

مثلا إذا طلبت منك أن تحسب لي مجموع عددين، فأنت تلقائيا تطبق خوارزمية تقوم بأخذ العددين وإجراء عملية الجمع عليهما ثم إعادة النتيجة، هذا باختصار هو مفهوم الخوارزميات.

ماهي أهمية الخوارزميات؟

في البرمجة، تعتبر خوارزميةً كل مجموعة أوامر قابلة للتحويل إلى لغة برمجية، والتي تؤدي دورا معينا في البرنامج، وتأتي مرحلة إعداد الخوارزمية مباشرة بعد مشكل أو حاجة، فحينما يطلب منك عميل إنجاز برنامج يعمل شغلا معينا، فهو يضع أمامك مشكلة بالنسبة لك، وحاجة بالنسبة له، وأنت بصفتك مبرمجا، عليك تحويل حاجة العميل من كلام بشري إلى

أوامر منطقية ومتسلسلة (أي إنشاء الخوارزمية المناسبة) ثم تحول هذه الخوارزمية إلى برنامج بواسطة إحدى لغات البرمجة، الشيء الذي يمكن تلخيصه في الشكل التالي:



بنية كتابة الخوارزميات

لإنشاء أية خوارزمية، فإنك ملزم بالتعرف على مجموعة من المفاهيم والتقنيات التي عليك لستخدامها من أجل وضع خوارزمية قابلة للتحويل إلى برنامج بشكل صحيح، وأولى هذه المفاهيم أن كل خوارزمية لها بداية ولها نهاية، وبين هذين الطرفين نكتب محتوى الخوارزمية الذي يؤدي إلى الحصول على النتيجة المتوخاة، لذلك وجب وضع خطوات أية خوارزمية بين الأمرين التاليين:

الشفرة بالعربية

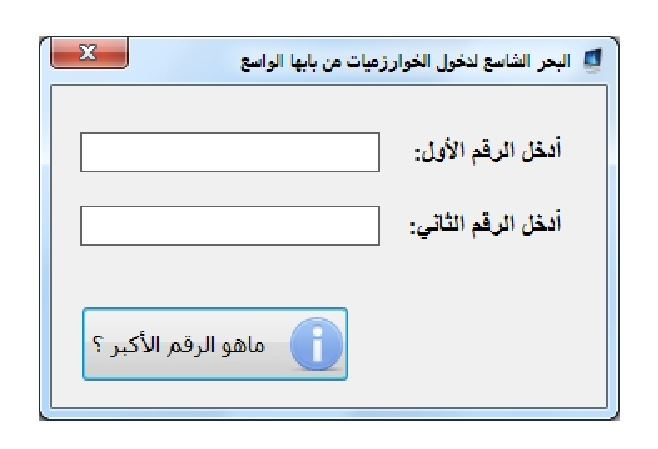
```
البداية
// هنا يكتب محتوى الخوارزمية
النهاية
```

الشفرة بالانجليزية

```
BEGIN
//Program Instructions
END
```

لنفترض أن العميل طلب منك برنامجا يقرأ رقمين، ثم يطبع أكبرهما (بمعنى أنه يريد منك أن تنشىء برنامجا يطلب من المستخدم أن يدخل رقمين كتابةً من لوحة المفاتيح أو من أي وسيلة إدخال أخرى، فيستقبل البرنامج هذين الرقمين ويقارنهما داخليا ويطبع على الشاشة Screen قيمة أكبر رقم).

كما تظهر الصورة التالية:



أول شيء ينبغي أن تفكر فيه، هو أنك ستحتاج إلى عددين للقيام بعملية المقارنة، وهذان العددان غير ثابتين و غير معروفين مسبقا، لأن المستخدم هو الذي سيدخلهما في البرنامج، وأنت لا يمكنك التكهن بهما، لذا فإنك ستحتاج إلى حاويات في الذاكرة تضع فيها القيم المدخلة لكي تقارنها في برنامجك، وللولوج إلى هذه الحاويات فأنت تحتاج إلى إعطائها أسماء مميزة وغير متشابهة، وكذلك تحتاج إلى تحديد نوع القيمة المراد تخزينها في كل حاوية.

القيم التي نجهلها ولا ندري قيمتها مسبقا، ونحتاج إلى تحديد نوع لها وتخزينها في الذاكرة تسمى متغيرات أو متحولات، أي أن قيمتها ليست ثابتة، وأنها قابلة للتغيير المستمر أثناء تنفيذ البرنامج.

إذن، فالمتغيرات هي قيم يحتاجها البرنامج الذي ننشؤه من أجل إنجاز عمليات معينة عليها، والمستخدم هو من يقوم بإدخال هذه القيم إلى البرنامج (عبر لوحة المفاتيح مثلا)، ويمكن للقيم المدخلة أن تكون من أنواع شتى، إما رقمية، أو نصية، أو عبارة عن تاريخ،...

الإعلان عن المتغيرات

الإعلان عن متغير يعني حجز مكان في الذاكرة من أجل تخزين قيمة معينة قابلة للتغير في زمن التنفيذ Runtime (المقصود بزمن التنفيذ الوقت الذي يكون البرنامج في شغالا)، ويكون بهذا الشكل:

الشفرة بالعربية

```
[نوع البيانات] اسم_المتغير؛
```

الشفرة بالانجليزية

```
[DATA_TYPE] VariableName ;
```

في الجزء الخاص بالكلمة [نوع البيانات] نضع نوع القيمة المراد تخزينها في المتغير مثلا : أرقام، حروف، نصوص (النص هو مجموعة حروف وأرقام)، نوع تاريخي (وقت أو تاريخ أو هما معا)، نوع منطقي (يقبل قيمتين متعارضتين وهما: صحيح أو خطأ)، وغيرها...

والجزء الخاص بالكلمة اسم_المتغير نضع الاسم الذي نريد إعطاءه للمتغير لكي نتمكن من استعماله في برنامجنا، والفاصلة المنقوطة في آخر السطر للدلالة على أن الأمر انتهى وأن السطر مكتمل، وهذه أمثلة لكيفية الإعلان عن مجموعة من المتغيرات:

الشفرة بالعربية

```
رقمي  الرقم_1؛              //الإعلان عن متغير رقمي اسمه الرقم_1
نصي  النص_1؛               //الإعلان عن متغير نصي اسمه النص_1
حرفي  حرف_1؛               //الإعلان عن متغير حرفي اسمه الحرف_1
تاريخي  تاريخ_الميلاد؛        //الإعلان عن متغير تاريخي اسمه تاريخ_الميلاد
منطقي  متغير_منطقي_1؛       //الإعلان عن متغير منطقي  اسمه متغير_منطقي_1
```

الشفرة بالانجليزية

```
Int Number1 ;              //Declare Integer Variable
String Text1 ;             //Declare  String Variable
Char Char1 ;               //Declare Char Variable
Date Date1 ;           //Declare Date Variable
Boolean Bool1 ;        //Declare Boolean Variable
```

في المثال أعلاه قمنا بالإعلان عن عدة متغيرات، لكننا لم نعطها قيما بدئية، بمعنى أن أماكن هذه المتغيرات في الذاكرة ما تزال فارغة null، ويلزمنا إعطاؤها قيما لكي نستخدمها، و هذه القيم إما أن تكون بدئية يحددها المبرمج في بداية الإعلان عن المتغيرات، أو تكون خلال مرحلة تنفيذ البرنامج بحيث تحصل المتغيرات على قيمها من خلال مدخلات المستخدمين.

أيضا يمكننا الاعلان عن مجموعة من المتغيرات ذات نفس نوع البيانات في سطر واحد، كما يعرض المثال الآتي:

الشفرة بالعربية

```
رقمي الرقم_1، الرقم_2 ، الرقم_3 ، الرقم_4؛
```

الشفرة بالانجليزية

```
Int Number1, Number2, Number3, Number4 ;
```

إسناد القيمة للمتغير

لإسناد قيمة بدئية لأي متغير، فإن الطريقة تكون عبر استخدام رمز السهم (→) ثم بعده القيمة المراد إعطاؤها للمتغير، ويمكن عمل ذلك في زمن الإعلان، وكذلك بعد الإعلان، و هذه أمثلة توضح هذا الكلام:

الشفرة بالعربية

```
رقمي الرقم_الأول ← 156؛      // إسناد قيمة للمتغير عند الإعلان عنه
رقمي الرقم_الثاني؛
الرقم_الثاني ← 48؛          //الإعلان عن المتغير أولا ثم إسناد قيمة له بعد ذلك
```

الشفرة بالانجليزية

```
Int First_Number ← 156 ;   // إسناد قيمة للمتغير عند الإعلان عنه
Int Second_Number ;
Second_Number ← 48 ;       //الإعلان عن المتغير أولا ثم إسناد قيمة له بعد ذلك
```

في السطر الأول أعطينا المتغير الرقم_الأول القيمة 156 في نفس وقت الإعلان، وفي السطر الثاني قمنا بإعطاء قيمة للمتغير الرقم_الثاني بعد الإعلان عنه، لاحظ أن المتغيرات الرقمية تستقبل القيم بشكل مباشر، بينما المتغيرات النصية فيلزمنا استخدام علامات التنصيص (" ") ووضع النص داخلهما، كما توضح الأمثلة الآتية:

الشفرة بالعربية

```
نصي الاسم_الكامل ← "خالد السعداني"؛   // إسناد قيمة للمتغير عند الإعلان عنه
نصي الجنسية؛
الجنسية ← "مغربي"؛                  //الإعلان عن المتغير أولا ثم إسناد قيمة له بعد ذلك
```

الشفرة بالانجليزية

```
String FullName ← "خالد السعداني";   // إسناد قيمة للمتغير عند الإعلان عنه
String Nationality ;
Nationality ← "مغربي";              //الإعلان عن المتغير أولا ثم إسناد قيمة له بعد ذلك
```

يمكننا إسناد قيمة متغير إلى متغير آخر بنفس الطريقة:

الشفرة بالعربية

```
رقمي الرقم_الأول ← 156؛
رقمي الرقم_الثاني؛
الرقم_الثاني ← الرقم_الأول؛
```

الشفرة بالانجليزية

```
Int First_Number ← 156 ;
Int Second_Number ;
Second_Number ← First_Number ;
```

قيمة هذا الأخير هي 156.

ولكي يسهل عليك معرفة من أين تنقل القيمة إلى أين اتبع دائما مسار السهم:

الرقم_الثاني ← الرقم_الأول؛ السهم يشير إلى أن قيمة المتغير الرقم_الأول ستنتقل إلى الرقم_الثاني.

إخراج البيانات:

ويقصد بإخراج البيانات عملية طباعة نص معين أو قيمة متغير على شاشة المستخدم، وفي الخوارزميات نستخدم الأمر: اكتب ونفتح القوسين، ثم نضع النص المراد إظهاره للمستخدم بين مزدوجتين، كما يعرض المثال التالي:

الشفرة بالعربية

```
// إظهار نص عادي
اكتب("السلام عليكم في عالم البرمجة")؛
// إظهار قيمة متغير
نص رسالة ← "الصبر مفتاح الفرج"؛
اكتب(رسالة)؛
```

الشفرة بالانجليزية

```
//إظهار نص عادي
WRITE("السلام عليكم في عالم البرمجة") ;
//إظهار قيمة متغير
String FullName ← "خالد السعداني";
WRITE(FullName) ;
```

قراءة المدخلات:

المقصود بقراءة المدخلات هو كيفية إدخال القيم المكتوبة من طرف المستخدم إلى البرنامج لكي نتعامل معها، كأن نكون بصدد القيام بعملية الجمع ونطلب منه أن يدخل رقمين، فعملية الإدخال هذه تحتاج منا إلى قراءة القيم داخل البرنامج لنعيد صياغتها وفق ما يحتاجه البرنامج منها، وتكون عملية قراءة المدخلات في الخوارزميات عبر الكلمة اقرأ ويتم كتابة القيمة أو المتغير الذي سيستقبل القيمة مباشرة بعدها بين قوسين، كما تعرض النماذج التالية:

الشفرة بالعربية

```
البداية
    نصي الاسم_الشخصي؛
    اكتب("رجاء أدخل اسمك الكريم: ")؛
    اقرأ(الاسم_الشخصي) ؛        // تخزين القيمة النصية المدخلة في المتغير المعلن عنه
    اكتب("مرحبا بك يا سيد: ")؛
    اكتب(الاسم_الشخصي)؛
النهاية
```

الشفرة بالانجليزية

```
BEGIN
    String FullName ;
    WRITE("رجاء أدخل اسمك الكريم") ;
    READ(FullName) ;        //تخزين القيمة النصية المدخلة في المتغير المعلن عنه
    WRITE(" مرحبا بك يا سيد:") ;
    WRITE(FullName) ;
END
```

في المثل أعلاه يظهر البرنامج رسالة يطلب فيها من المستخدم إدخال اسمه الشخصي، وحينما يتم كتابة الاسم، يتم قراءته وتخزينه في المتغير المسمى الاسم_الشخصي ، وبعد ذلك يعيد البرنامج طباعة الاسم مسبوقا بعبارة مرحبا بك يا سيد.:

الروابط / المعاملات:

الروابط أو المعاملات هي رموز نستخدمها لإجراء بعض العمليات على المتغيرات أو على القيم بصفة عامة مثل العمليات الحسابية، أو عمليات مقارنة القيم (تحديد القيمة الكبرى والقيمة الصغرى) وغير ذلك.

الروابط الحسابية أو الرياضية Arithmetic operators:

وهذا جدول رموز الروابط وأمامها الدور المنوط بها:

الرابط	دوره
+	الجمع
-	الطرح
*	الجداء
/	القسمة
\	القسمة الصحيحة الطبيعية
%	باقي القسمة
^	القوة

الروابط الظاهرة أعلاه هي رموز نستخدمها حينما نود القيام بعملية حسابية على قيمتين أو أكثر، وهذه أمثلة على استخدام الروابط أعلاه:

الشفرة بالعربية

```
البداية
    رقمي الرقم_الأول، الرقم_الثاني؛
    الرقم_الأول ← 25؛
    الرقم_الثاني ← 5؛

    // استخدام الروابط لحساب قيم المتغيرين السابقين

    //الجمع
    رقمي الجمع ← الرقم_الأول + الرقم_الثاني؛

    //الطرح
    رقمي الطرح ← الرقم_الأول - الرقم_الثاني؛

    //الضرب
    رقمي الضرب ← الرقم_الأول * الرقم_الثاني؛

    //القسمة
    رقمي القسمة ← الرقم_الأول / الرقم_الثاني؛

    //القوة Power
    رقمي القوة ← الرقم_الأول ^ الرقم_الثاني؛

    //باقي القسمة Modulo
    رقمي الباقي ← الرقم_الأول % الرقم_الثاني؛
النهاية
```

الشفرة بالانجليزية

```
BEGIN
      Int First_Number, Second_Number ;
      First_Number ← 25 ;
      Second_Number  ← 5 ;

      //استخدام الروابط لحساب قيم المتغيرين السابقين

      //الجمع
      Int Sum  ← First_Number + Second_Number ;

      //الطرح
      Int Substract  ← First_Number - Second_Number ;

      //الضرب
      Int Multiplication  ← First_Number * Second_Number ;

      //القسمة
      Int Division  ← First_Number / Second_Number ;

      // القوةPower
      Int Power  ← First_Number ^ Second_Number ;

      // باقي القسمةModulo
      Int Modulo  ← First_Number % Second_Number ;

END
```

المتغير **الجمع** يقوم **بجمع** قيمتي المتغيرين **الرقم_الأول** و **الرقم_الثاني** من خلال استخدام الرابط +، أي أن قيمته ستكون هي 25 + 5 أي 30.

المتغير **الطرح** يقوم **بطرح** قيمة المتغير **الرقم_الأول** من قيمة المتغير **الرقم_الثاني** عبر استخدام علامة ناقص - ، أي أن قيمته ستكون هي 25 - 5 أي 20.

المتغير الضرب يقوم بضرب قيمة المتغير الرقم_الأول في قيمة المتغير الرقم_الثاني عبر استعمال الرابط *، أي أن قيمته ستكون هي 25 * 5 أي 125.

المتغير القسمة يقوم بقسمة قيمة المتغير الرقم_الأول على المتغير الرقم_الثاني عبر استخدام رمز القسمة، أي أن النتيجة ستكون هي 25 / 5 أي 5.

المتغير القوة يقوم بحساب الأساس الرقم_الأول ذي الأس الرقم_الثاني أي أن النتيجة ستكون هي 255، أي 9765625.

المتغير الباقي يقوم بحساب باقي قسمة قيمة المتغير الرقم_الأول على قيمة المتغير الرقم_الثاني ، وكما هو معلوم فباقي قسمة 25 على 5 هو صفر لأنه لا يتبقى شيء.

روابط دمج النصوص String Concatenation operators:

ويمكننا هذا النوع من الروابط من دمج نصين أو أكثر مع بعضهم البعض، ويكون من خلال استخدام الرمز +، وهذا مثال على دمج قيمتين نصيتين:

الشفرة بالعربية

```
البداية
    نصي النص_الأول، النص_الثاني؛
    النص_الأول ← "لا يجتمع سيفان"؛
    النص_الثاني ← "في غِمْدٍ واحد"؛
    نصي الدمج ← النص_الأول + النص_الثاني؛
    اكتب(الدمج)؛
النهاية
```

الشفرة بالانجليزية

```
BEGIN
String Text1, Text2 ;

Text1 ← "لا يجتمع سيفان" ;

Text2 ← "في غِمْدٍ واحد" ;

String Concat  ← Text1 + Text2;

WRITE(Concat)

END
```

الآن صارت القيمة لا يجتمع سيفان في غمد واحد هي القيمة المخزنة في المتغير الدمج بعد دمج قيمتي المتغيرين الأولين بواسطة رمز الدمج +.

روابط الزيادة والقصان Increment and Decrement Operators:

وهي روابط نستخدمها مع المتغيرات الرقمية من أجل زيادة قيمتها بواحد أو إنقاص قيمتها بواحد، وصيغة الرابطين كالتالي:

الشفرة بالعربية

```
البداية
    //زيادة قيمة المتغير ب 1

    رقمي  الرقم_الأول ← 25؛
    الرقم_الأول ← الرقم_الأول +1؛
    //أو هكذا:
    الرقم_الأول ++؛

    // نقصان قيمة المتغير ب 1

    رقمي  الرقم_الثاني ← 25؛
    الرقم_الثاني ← الرقم_الثاني -1؛
    //أو هكذا:
    الرقم_الثاني --؛
النهاية
```

الشفرة بالانجليزية

```
BEGIN
        //زيادة قيمة المتغير ب 1//

        Int First_Number ← 25 ;
        First_Number ← First_Number + 1;
        //أو هكذا://
        First_Number++;
        //نقصان قيمة المتغير ب 1//

        Int Second_Number ← 25 ;
        Second_Number ← Second_Number - 1;
        //أو هكذا://
        Second_Number--;
END
```

وهي روابط نستخدمها من أجل مقارنة قيمتين وتحديد نوع العلاقة بينهما (أكبر من، أصغر من، تساوي، ...) ونتيجة المقارنة تكون منطقية boolean، أي لا تقبل إلا قيمتين: إما صحيح true وإما خطأ false، فيما يلي جدول يعرض رموز الروابط المستخدمة في عمليات المقارنة:

الرابط	دوره
>	أكبر من
<	أصغر من
=	يساوي
<>	يخالف
>=	أكبر من أو يساوي
<=	أصغر من أو يساوي

وهذه أمثلة على استخدام هذا النوع من الروابط:

الشفرة بالعربية

```
البداية
    // العبارة الأولى ستحتوي على النتيجة صحيح True لأن 4 أكبر من 2
    منطقي العبارة_الأولى ← 2 > 4؛

    // العبارة الثانية ستحتوي على النتيجة خطأ False لأن 5 ليست أصغر من 1
    منطقي العبارة_الثانية ← 1 < 5؛

    // العبارة الثالثة ستحتوي على القيمة صحيح True لأن 5 تخالف 10
    منطقي العبارة_الثالثة ← 5 <> 10؛

    // العبارة الرابعة ستحتوي على القيمة صحيح True لأن 20 تساوي مجموع 6 و 14.
    منطقي العبارة_الرابعة ← 6 + 14 = 20؛
النهاية
```

الشفرة بالانجليزية

```
BEGIN
    // العبارة الأولى ستحتوي على النتيجة صحيح True لأن 4 أكبر من 2
    Bool Expression1 ← 4 > 2 ;

    // العبارة الثانية ستحتوي على النتيجة خطأ False لأن 5 ليست أصغر من 1
    Bool Expression2 ← 5 < 1 ;
    // العبارة الثالثة ستحتوي على القيمة صحيح True لأن 5 تخالف 10
    Bool Expression3 ← 10 <> 5 ;

    // العبارة الرابعة ستحتوي على القيمة صحيح True لأن 20 تساوي مجموع 6 و 14.
    Bool Expression4 ← 20 = 14 + 6 ;
END
```

الروابط المنطقية Logical operators:

هي روابط نستخدمها من أجل الحصول على نتيجة شرطين أو أكثر، والنتيجة تكون منطقية boolean، أي لا تقبل إلا قيمتين: إما صحيح true وإما خطأ false، هذه الروابط هي AND والذي يعني (و) وتكون النتيجة صحيحة إذا كانت كل أطراف الشرط صحيحة، والرابط OR والذي يعني (أو) وتكون النتيجة صحيحة إذا كان هنالك طرف واحد فقط أو أكثر في الشرط صحيحا.

حتى نبسط مفهوم الروابط المنطقية، دعنا نستحضر مثالا سهلا، لو قلت لك: سيزورني أحمد و كمال، فذلك يعني أن كلامي سيكون صحيحا حينما سيأتيان معا، لكن لو قلت لك: قد يزورني أحمد أو كمال، فالمعنى مختلف وكلامي سيكون صحيحا سواء حضر أحمد أو كمال.

الآن لاحظ معي هذا المثال لتتعرف على كيفية استخدام الروابط المنطقية:

```
البداية
// العبارة الأولى ستحتوي على النتيجة صحيح True لأن 2 أصغر من 4، و 7 أكبر من 5
مضقي العبارة_الأولى ← (4 < 2 و 5 < 7)؛

// العبارة الثانية ستحتوي على النتيجة خطأ False لأن أحد طرفي المعادلة غير صحيح لأن 3 لا تساوي 5
مضقي العبارة_الثانية ← (4 < 2 و 5 = 3)؛

/* العبارة الثالثة
ستحتوي على النتيجة صحيح True
لأنه يوجد شرط واحد على الأقل صحيح وهو 4 < 2
وبما أننا نستخدم الرابط " أو " فيكفي وجود شرط
واحد صحيح لتكون النتيجة صحيحة.
*/
مضقي العبارة_الثالثة ← (4 < 2 أو 5 < 2)؛

// العبارة الرابعة ستحتوي على النتيجة خطأ لأن طرفي المعادلة غير صحيحين معا
مضقي العبارة_الرابعة ← (4 > 2 أو 5 = 3)؛
النهاية
```

```
BEGIN
        // لأن 2 أصغر من 4، و 7 أكبر من 5 True العبارة الأولى ستحتوي على النتيجة صحيح

        Bool Expression1 ← ( 2 < 4 AND 7 > 5 ) ;

        // لأن أحد طرفي المعادلة غير صحيح لأن 3 لا تساوي 5 False العبارة الثانية ستحتوي على النتيجة خطأ

        Bool Expression2 ← ( 2 < 4 AND 3 = 5 ) ;

        /* العبارة الثالثة
         True ستحتوي على النتيجة صحيح
         لأنه يوجد شرط واحد على الأقل صحيح وهو 2 < 4
        وبما أننا نستخدم الرابط " أو " فيكفي وجود شرط
        واحد صحيح لتكون النتيجة صحيحة.
        */

        Bool Expression3 ← ( 2 > 5 OR 2 < 4 ) ;

        // العبارة الرابعة ستحتوي على النتيجة خطأ لأن طرفي المعادلة غير صحيحين معا

        Bool Expression4 ← ( 2 > 4 OR 3 = 5 ) ;

END
```

البنية الشرطية:

أحيانا نحتاج إلى إجراء تحقق من عبارة معينة في برنامجنا، فإن كانت نتيجة التحقق تساوي قيمة معينة نفذ شفرة معينة، وإن كان العكس أو كانت تساوي قيمة أخرى نفذ شفرات أخرى، على سبيل المثال، نريد إنشاء برنامج يتكون من شاشة لتسجيل الدخول مثل تلك التي في برنامج Skype أو برنامج Messenger، فنحن ملزمين بالتحقق من اسم المستخدم وكلمة السر المدخلين، إن كانا صحيحين سمحنا بعملية الدخول، وإن كان أحدهما خاطئا أظهرنا رسالة خطأ.

في البرمجة تسمى عملية التحقق من عبارة معينة شرطا Condition أو بنية شرطية Flow Control، وصيغتها كما يلي:

الشفرة بالعربية

```
البداية

    إذا كان ( // العبارة الشرطية) إذن

        // إذا تحققت العبارة الشرطية سيتم تنفيذ هذه الأوامر

    أما إذا كان ( //عبارة شرطية أخرى) إذن

        // إذا تحققت العبارة الشرطية سيتم تنفيذ هذه الأوامر

    وإلا
        /* في حال عدم تحقق أي شرط
        من الشروط السابقة فسيتم تنفيذ هذا الجزء الموجود في الجزء " وإلا "     */

    نهاية إذا كان
النهاية
```

الشفرة بالانجليزية

```
BEGIN

      IF ( /*Statement that can be either true or false*/) THEN
            //Do Someting
      ELSEIF (/* Other Condition */) THEN
            //Do Someting
      ELSE
            //Do Someting
      END IF

END
```

الآن لنعطي مثالا نتعلم من خلاله طريقة لستخدام البنية الشرطية في برامجنا لنفترض أننا بصدد إنشاء شاشة لتسجيل الدخول، مثل الشاشة الظاهرة في الصورة التالية:



في شاشة الدخول أعلاه، سيقوم المستخدم بكتابة اسمه، وبكتابة كلمة المرور، ونحن سنستقبل هذين القيمتين، وسنقوم بتخزينهما في متغيرين نصيين، ثم نقارنهما مع البيانات المسجلة

عندنا في البرنامج أو المخزنة في ملف أو قاعدة بيانات، فإن كان هناك توافق بين البيانات المدخلة وبين البيانات المخزنة، نسمح بعملية الدخول، وإلا نظهر رسالة خطأ تعلم المستخدم بأن عملية تسجيل الدخول فشلت.

وهذه هي الخوارزمية التي علينا وضعها بالاعتماد على البنية الشرطية لإنجاز محاكاة لهذا البرنامج:

الشفرة بالعربية

```
البداية
    نصي اسم_المستخدم_المخزن ← "myUserName"؛
    نصي كلمة_المرور_المخزنة ← "MyPassword123"؛
    نصي اسم_المستخدم؛
    نصي كلمة_المرور؛

    اكتب("أدخل اسم المستخدم من فضلك")؛

    اقرأ (اسم_المستخدم)؛

    اكتب("أدخل كلمة المرور من فضلك")؛

    اقرأ (كلمة_المرور)؛

    إذا كان (اسم_المستخدم = اسم_المستخدم_المخزن و كلمة_المرور = كلمة_المرور_المخزنة) إذن
    اكتب("تم تسجيل الدخول بنجاح")؛

    أما إذا كان (اسم_المستخدم <> اسم_المستخدم_المخزن و كلمة_المرور = كلمة_المرور_المخزنة) إذن
    اكتب("اسم المستخدم المدخل غير صحيح")؛

    أما إذا كان (اسم_المستخدم = اسم_المستخدم_المخزن و كلمة_المرور <> كلمة_المرور_المخزنة) إذن
    اكتب("كلمة المرور غير صحيحة")؛

    وإلا
    اكتب("عفوا، البيانات المدخلة غير صحيحة")؛

    نهاية إذا كان
النهاية
```

```
BEGIN
      String SavedID ← " myUserName";
      String SavedPWD ← "  MyPassword123";
      String ID ;
      String Password ;

      WRITE("أدخل اسم المستخدم من فضلك");

      READ(UserName);

      WRITE(" أدخل كلمة المرور من فضلك");

      READ(UserName);

      IF (ID = SavedID AND Password= SavedPWD) THEN

            WRITE(" تم تسجيل الدخول بنجاح");

      ELSEIF (ID <> SavedID AND Password= SavedPWD) THEN

            WRITE(" اسم المستخدم المدخل غير صحيح");

      ELSEIF (ID = SavedID AND Password<> SavedPWD) THEN

            WRITE(" كلمة المرور غير صحيحة");

      ELSE (ID = SavedID AND Password<> SavedPWD)

            WRITE(" عفوا، البيانات المدخلة غير صحيحة");

      END IF
END
```

تمارين البنية الشرطية:

التمرين الأول: أنشيء برنامجا يطلب من المستخدم إدخال قيمتين رقميتين، ثم يقارنهما ويطبع أكبرهما.

التمرين الثاني: أنشيء برنامجا يطلب من المستخدم إدخال قيمة رقمية ثم يعيد له هل القيمة المدخلة موجبة أم سالبة.

التمرين الثالث: أنشيء برنامجا يستقبل ثلاثة قيم ثم يقوم بطباعة أكبرها.

التمرين الرابع: أنشيء برنامجا يستقبل رقمين ثم يعيد لنا هل نتيجة عملية ضربهما موجبة أم سالبة.

التمرين الخامس: أنشيء برنامجا يطلب من المستخدم إدخال قيمة نصية، فإن كانت مساوية للكلمة "JAVA" أو الكلمة "PERL" يظهر له رسالة مفادها أن الجواب صحيح وإن كانت على خلاف ذلك يظهر له رسالة مفادها أن الجواب خاطيء.

التمرين السادس: أنشيء برنامجا يحسب المبلغ العام لمنتوج معين (علما أن المبلغ العام = ثمن الوحدة الواحدة من المنتوج * الكمية المراد بيعها) لكن قبل ذلك يتم التحقق من الكمية إن كانت أصغر من أو تساوي صفر يتم إظهار رسالة خطأ، وإن كانت أكبر قطعا من صفر يتم حساب المبلغ العام.

التمرين السابع: أنشيء برنامجا يقوم بحساب المبلغ العام لمنتوج معين بالأخذ بعين الاعتبار نسبة الخصم:

1. إن كان المبلغ العام أصغر من 500 درهم فإن نسبة الخصم صفر.

2. إن كان المبلغ العام أكبر من 500 وأصغر من 1000 فإن نسبة الخصم هي 1 في المئة.

3. إن كان المبلغ العام أكبر قطعا من 1000 درهم فإن نسبة الخصم هي 5 في المئة.

علما أن المبلغ النهائي يحسب وفقا للصيغة التالية:

المبلغ النهائي = المبلغ * (1 ـ نسبة الخصم / 100)

التمرين الثامن: أنشىء برنامجا يطلب من المستخدم إدخال الكمية المراد بيعها والكمية الموجودة في المخزن، ثم يتحقق من أن الكمية المراد بيعها أصغر من الكمية المخزنة لكي يسمح بعملية البيع، وإلا فإنه يظهر رسالة خطأ مفادها أن الكمية المراد بيعها غير متوفرة كاملة في المخزن.

البنية التكرارية Loops

أحيانا نحتاج إلى تكرار أمر برمجي معين أكثر من مرة داخل تطبيقاتنا، فلو افترضنا أننا بصدد برمجة تطبيق صغير يخزن قيمة رقمية في متغير معين ويطلب من المستخدم تخمين هذه القيمة، حتما نحن لا نعرف عدد الاحتمالات التي سيقوم بها المستخدم لكي يصل إلى نفس القيمة المخزنة عندنا، لذلك يتوجب طرح نفس السؤال في كل مرة يدخل المستخدم فيها قيمة مخالفة للقيمة المخزنة.

أو لنفترض أننا بصدد إنتاج برنامج يسأل المستخدم عن اسم أول خلفاء الدولة العباسية، فإن كان الجواب صائبا يتوقف البرنامج وإن كان الجواب خاطئا يعيد طرح السؤال على المستخدم إلى حين يحصل على الجواب الصحيح، في حالتنا هذه نحن لا نعلم عدد المرات اللازمة لكي يجيب المستخدم بالجواب الصائب، لذلك فنحن بحاجة إلى إجراء تكرار للأمر الذي يطبع السؤال في كل مرة يخطىء المستخدم في الإجابة.

لإجراء تكرار أمر معين في لغات البرمجة توجد أكثر من صيغة، سنتناول نحن في هذا الجزء أشهر صيغتين وهما الصيغة التكرارية الشرطية ما دام while والصيغة التكرارية الحسابية لأجل for، ولنبدأ بالصيغة الأولى ولنطبقها على نفس المثال الأخير المتعلق بأول خلفاء الدولة العباسية.

الصيغة التكرارية الشرطية : ما دام while:

وهي صيغة تبدأ بشرط مسبوق بالكلمة ما دام، وتعني أنه مادام هذا الشرط متحققا فقم بتكرار الأوامر التي تأتي بعده، وحينما لا يتحقق الشرط يتم الخروج من البنية التكرارية والانتقال إلى ما بعدها.

وهذه هي صيغة البنية التكرارية باستخدام الكلمة ما دام:

الشفرة بالعربية

```
البداية
    ما دام (/* الشرط */)
        // نفذ ما يلي
    نهاية ما دام
النهاية
```

الشفرة بالانجليزية

```
BEGIN
      WHILE (Expression)
            //Statements
      END WHILE
END
```

خوارزمية تخمين أول خلفاء الدولة العباسية ستكتب بهذا الشكل إذن:

الشفرة بالعربية

```
البداية
    نصي اسم_الخليفة_الأول ؛

    اكتب("ما هو اسم أول خليفة في الدولة العباسية؟ ")؛

    اقرأ (اسم_الخليفة_الأول)؛

    ما دام (اسم_الخليفة_الأول <> " أبو العباس السفاح ")
        اكتب("الأسف لم تسجل الإجابة الصحيحة ! ")؛

    نهاية ما دام
    اكتب("ممتاز ! لقد أدخلت الجواب الصائب.")؛

النهاية
```

الشفرة بالانجليزية

```
BEGIN
	String First_Khalifa ;
	WRITE("ما هو اسم أول خليفة في الدولة العباسية ؟") ;
	READ(First_Khalifa) ;
	WHILE First_Khalifa <> " أبو العباس السفاح "
		WRITE("للأسف لم تسجل الإجابة الصحيحة ! ") ;
	END WHILE
	WRITE("ممتاز ! لقد أخذت الجول الصائب.")
END
```

ملاحظة:

قد تتساءل: ولماذا لا نستخدم الجملة الشرطية إذا كان مباشرة، سأجيبك: بأن الكلمة إذا كان تتعامل مع القيمة المدخلة لمرة واحدة ثم تنتقل إلى ما بعدها، بينما البنية التكرارية ستعيد نفس الأمر ما دام الشرط متحققا.

الصيغة التكرارية الحسابية: لأجل for:

الصيغة التكرارية الحسابية مهمة جدا في البرمجة، ونحتاجها كثيرا في تطبيقاتنا لأنها تساعد على تكرار أمر معين بالعدد الذي نشاء، فتخيل لو أننا نريد طباعة جملة **الصدق عز والكذب ذل وخضوع** ألف مرة كم سيلزمنا من سطر من الجملة **الصدق عز والكذب ذل وخضوع**، أكيد سنحتاج إلى ألف سطر، وهذا شيء متعب ومرهق جدا للمبرمج، لذلك فالبنية التكرارية

الحسابية تسهل علينا هذا الأمر بشكل رائع جدا، بحيث يكفي أن نحدد بداية التكرار ونهايته ثم نكتب الأمر المراد تكراره مرة واحدة ويتم إعادته حسب المجال الرقمي الذي حددناه.

صيغة البنية التكرارية الحسابية تكتب بهذا الشكل:

الشفرة بالعربية

```
البداية
رقمي العداد؛
    لأجل العداد ← 1 إلى 1000
        اكتب (" الصدق عز والكذب ذل وخضوع ")
    نهاية لأجل
النهاية
```

الشفرة بالانجليزية

```
BEGIN
Int Count ;
      FOR Count  ← 1 TO 1000
            WRITE (" الصدق عز والكذب ذل وخضوع ") ;
      END FOR
END
```

بعد تنفيذ الخوارزمية أعلاه سيتم طباعة الجملة المذكورة ألف مرة لأن التكرار الحسابي يبدأ من 1 وينتهي بألف.

وهذا مثال آخر يطلب من المستخدم أن يدخل رقما، ثم يقوم بجمع هذا الرقم مع جميع الأرقام التي تسبقه، ويطبع النتيجة، مثلا لو أدخل المستخدم الرقم 6 فسوف يقوم البرنامج بالعملية التالية:

$$21 = 6 + 5 + 4 + 3 + 2 + 1$$

الشفرة بالعربية

```
البداية
رقمي الرقم، العداد؛
رقمي المجموع ← 0؛
اكتب ("من فضلك أدخل رقما ")
اقرأ (الرقم)
لأجل العداد ← 1 إلى الرقم
المجموع ← المجموع + الرقم ؛
نهاية لأجل
اكتب ("المجموع يساوي: " + المجموع)
النهاية
```

الشفرة بالانجليزية

```
BEGIN
      Int Number, Count ;
      Int Sum  ← 0 ;
      WRITE (" من فضلك أدخل رقما") ;
      READ(Number) ;
      FOR Count  ←  1 TO Number
            Sum  ←  Sum + Number ;
      END FOR
      WRITE(" :المجموع يساوي" + Sum) ;
END
```

قمنا بالإعلان عن متغيرين رقميين اسمهما: **الرقم** و هو القيمة الرقمية التي سيدخلها المستخدم، والمتغير: العداد و هو الذي سنستخدمه في التكرار بحيث سيبدأ من الرقم 1 وينتهي بالرقم المدخل من طرف المستخدم، وفي كل تكرار يضيف قيمة الرقم الذي يصل إليه إلى قيمة المتغير الذي أسميناه: المجموع.

لنفترض أن المستخدم قام بإدخال الرقم 7، بناء على ذلك فعملية التكرارستوزع القيم على المتغيرات كما يعرض الجدول التالي:

الرقم	العداد	المجموع
7	1	1

7	2	3=2+1
7	3	6=3+3
7	4	10=4+6
7	5	15=5+10
7	6	21=6+15
7	7	28=7+21

المصفوفات Arrays

خلال الفقرات السابقة كنا نتعامل مع المتغيرات، ورأينا كيف تسمح لنا بتخزين القيم في الذاكرة، ورأينا كذلك أن كل متغير مخصص لاحتواء قيمة واحدة، ولا يمكن للمتغير الواحد أن يستقبل أكثر من قيمة في وقت واحد.

لذلك لو افترضنا أننا نريد إنشاء برنامج لحساب معدلات الطلبة، بحيث يستقبل هذا البرنامج نقطة الطالب في كل مادة من المواد التي يدرسها، ثم يقوم بحساب المعدل العام، قد تقول لي: بإمكاننا عمل ذلك من خلال الإعلان عن مجموعة من المتغيرات حسب عدد المواد لدينا ثم نجري عليها عملية حساب المعدل العام، سأقول لك بأن جوابك صائب، لأنه لو أردنا مثلا حساب معدل طالب معين في عشرة مواد يدرسها، سوف نعلن عن عشر متغيرات رقمية لاستقبال نقط الطالب، بالطريقة التالية:

الشفرة بالعربية

```
رقمي الرقم1، الرقم2، الرقم3، الرقم4، الرقم5، الرقم6، الرقم7، الرقم8، الرقم9، الرقم10؛
```

الشفرة بالانجليزية

```
Int Num1, Num2, Num3, Num4, Num5, Num6, Num7, Num8, Num9, Num10;
```

لكن هذه الطريقة ليست مجدية إذا كنا سنحتاج إلى تخزين قيم كثيرة، لأننا سنجد صعوبة في استحضار أسماء المتغيرات، ناهيك عن صعوبة تتبع الكود وقراءته.

لحل هذه المشاكل وغيرها وجد مفهوم المصفوفات Arrays، بحيث نستخدمها حينما نود تخزين مجموعة من القيم المنتمية لنفس نوع البيانات في متغير واحد، ويرتكز مفهوم المصفوفات على ركيزتين أساسيتين هما:

القيمة Value: وهي القيم المراد تخزينها في عناصر المصفوفة، لو أخذنا مثلا مصفوفة لتخزين درجات الطلاب في المواد فإن الدرجات هي القيم، كل درجة عبارة عن قيمة سيتم تخزينها في عنصر معين من عناصر مصفوفة المواد.

الرتبة Index: وهي رتبة العنصر داخل المصفوفة، وتبدأ بصفر وتنتهي برتبة آخر عنصر ناقص واحد، مثلا لو أردنا تخزين الدرجات في مصفوفة المواد فإن التمثيل الفعلي سيكون بالشكل التالي:

نستطيع الوصول إلى أي عنصر من عناصر المصفوفة من خلال رتبته Index، هذا النوع من المصفوفات الذي نتحدث عنه يسمى المصفوفات الأحادية البعد one-dimensional array لأنها تحتوي على بعد واحد يضم العناصر بشكل خطي كما يعرض الشكل أعلاه.

الإعلان عن مصفوفة أحادية:

الشفرة بالعربية

```
البداية
    //مصفوفة من دون تحديد عدد العناصر
    نوع_المصفوفة اسم_المصفوفة_الأولى [ ]؛

    //مصفوفة مكونة من 7 عناصر
    نوع_المصفوفة اسم_المصفوفة_الثانية [7]؛
النهاية
```

الشفرة بالانجليزية

```
BEGIN
      //مصفوفة من دون تحديد عدد العناصر
      Data_Type ArrayName[] ;

      //مصفوفة مكونة من 7 عناصر
      Data_Type ArrayName[7] ;
END
```

أمثلة على إعلان المصفوفات أحادية البعد:

الشفرة بالعربية

```
البداية
    //مصفوفة من دون تحديد عدد العناصر
    رقمي مصفوفة_درجات_الطلاب [ ]؛

    //مصفوفة مكونة من 7 عناصر
    رقمي مصفوفة_درجات_الطلاب [7] ؛
النهاية
```

الشفرة بالانجليزية

```
BEGIN
        //مصفوفة من دون تحديد عدد العناصر
        Int MarksArray[] ;

        //مصفوفة مكونة من 7 عناصر
        Int MarksArray[7] ;
END
```

مثل عملي على كيفية استخدام المصفوفات:

الشفرة بالعربية

```
البداية
        //مصفوفة مكونة من 7 عناصر لتخزين أيام الأسبوع
        نصي مصفوفة_أيام_الأسبوع [7] ؛

        //متغير من نوع رقمي للقيام بعملية تكرار تذهب من أول عنصر إلى آخر عنصر في المصفوفة
        رقمي  العداد ؛

        //تخزين القيم في عناصر المصفوفة حسب الرتبة Index
        مصفوفة_أيام_الأسبوع [0] ← "الأحد"؛
        مصفوفة_أيام_الأسبوع [1] ← "الإثنين"؛
        مصفوفة_أيام_الأسبوع [2] ← "الثلاثاء"؛
        مصفوفة_أيام_الأسبوع [3] ← "الأربعاء"؛
        مصفوفة_أيام_الأسبوع [4] ← "الخميس"؛
        مصفوفة_أيام_الأسبوع [5] ← "الجمعة"؛
        مصفوفة_أيام_الأسبوع [6] ← "السبت"؛

        //عرض كل عناصر المصفوفة
        لأجل العداد ← 0 إلى 6
            اكتب ("اسم اليوم: " + مصفوفة_أيام_الأسبوع [العداد]) ؛
        نهاية لأجل
النهاية
```

```
BEGIN
      //مصفوفة مكونة من 7 عناصر لتخزين أيام الأسبوع
      String WeekArray[7] ;

      //متغير من نوع رقمي للقيام بعملية تكرار تذهب من أول عنصر إلى آخر عنصر في المصفوفة
      Int Count ;
      // تخزين القيم في عناصر المصفوفة حسب الرتبة
      WeekArray[0] ← " Sunday";
      WeekArray[1] ← " Monday";
      WeekArray[2] ← " Tuesday";
      WeekArray[3] ← " Wednesday";
      WeekArray[4] ← " Thursday";
      WeekArray[5] ← " Friday";
      WeekArray[6] ← " Saturday";
      //عرض كل عناصر المصفوفة
      For Count ← 0 TO 6
            WRITE("The Day Name Is: "+ WeekArray[Count]);
      END FOR
END
```

المصفوفات متعددة الأبعاد

المصفوفات متعددة الأبعاد هي مصفوفات تتكون من أكثر من بعد، قد تكون ثنائية البعد (عبارة عن أعمدة وأسطر) أو ثلاثية البعد أو أكثر من ذلك حسب الحاجة، علما أن هذا النوع من المصفوفات نادر الاستخدام جدا ولا نحتاجه إلا في حالة رغبتنا في التعمق في برمجة الألعاب أو برامج التصميم...

فيما يلي مثل عن الإعلان عن مصفوفة ثنائية two-dimensional array من نوع رقمي تتكون من خمسة أعمدة Columns وأربعة أسطر Rows:

الشفرة بالعربية

```
رقمي مصفوفة_ثنائية [5,4]؛
```

الشفرة بالانجليزية

```
Int Two_Dimensional_Array[4,5];
```

المصفوفة أعلاه عبارة عن مصفوفة رقمية ثنائية البعد تتكون من أربعة أسطر وخمسة أعمدة، إذا أردنا تمثيلها رياضيا فهي بمثابة جدول بالشكل التالي:

(0,0)	**(0,1)**	**(0,2)**	**(0,3)**	**(0,4)**
(1,0)	**(1,1)**	**(1,2)**	**(1,3)**	**(1,4)**
(2,0)	(2,1)	(2,2)	(2,3)	**(2,4)**
(3,0)	**(3,1)**	**(3,2)**	**(3,3)**	**(3,4)**

ملحوظة: انتبه إلى أن ترتيب العناصر Index يبدأ من صفر سواء بالنسبة للأعمدة أو الأسطر، وينتهي دائما برتبة آخر عنصر في المصفوفة ناقص واحد، وعدد عناصر المصفوفة الثنائية البعد يساوي عدد الأعمدة في عدد الأسطر، في حالتنا أعلاه فعدد عناصر مصفوفة مكونة من خمسة أعمدة وأربعة أسطر هو عشرون خانة لاحتواء قيم العناصر.

مثال على استخدام المصفوفات المتعددة البعد:

فيما يلي مثال برمجي يوضح كيفية الإعلان عن المصفوفات الثنائية البعد وتعبئة عناصرها بالقيم:

الشفرة بالعربية

```
البداية
    //مصفوفة مكونة من عمودين وثلاثة أسطر
    رقمي مصفوفة_ثنائية [2,3] ؛

    //متغير من نوع رقمي للقيام بعملية تكرار تذهب من أول عنصر إلى آخر عنصر في المصفوفة
    رقمي   عداد_الأعمدة، عداد_الأسطر ؛

    //تخزين القيم في عناصر المصفوفة حسب الرتبة Index
    مصفوفة_ثنائية [0,0] ← 18؛
    مصفوفة_ثنائية [0,1] ← 14؛
    مصفوفة_ثنائية [0,2] ← 16؛

    مصفوفة_ثنائية [1,0] ← 11؛
    مصفوفة_ثنائية [1,1] ← 19؛
    مصفوفة_ثنائية [1,2] ← 20؛


    //عرض كل عناصر المصفوفة
    لأجل عداد_الأسطر ← 0 إلى 2
        لأجل عداد_الأعمدة ← 0 إلى 1
            اكتب ("قيمة العنصر هي: " +مصفوفة_ثنائية[عداد_الأسطر، عداد_الأعمدة]) ؛
        نهاية لأجل
    نهاية لأجل

النهاية
```

```
BEGIN
      //مصفوفة مكونة من عمودين وثلاثة أسطر
      Int Two_Dimensional_Array[3,2];

      //متغير من نوع رقمي للقيام بعملية تكرار تذهب من أول عنصر إلى آخر عنصر في المصفوفة
      Int Count_Rows, Count_Columns ;

      // تخزين القيم في عناصر المصفوفة حسب الرتبة
      Two_Dimensional_Array[0,0] ← 18 ;
      Two_Dimensional_Array[0,1] ← 14 ;
      Two_Dimensional_Array[0,2] ← 16 ;

      Two_Dimensional_Array[1,0] ← 11 ;
      Two_Dimensional_Array[1,1] ← 19 ;
      Two_Dimensional_Array[1,2] ← 20 ;


      //عرض كل عناصر المصفوفة
      For Count_Columns ← 0 TO 2
            For Count_Rows ← 0 TO 1
                  WRITE("The Element Is: "+
                  Two_Dimensional_Array[Count_Columns, Count_Rows]);
            END FOR
      END FOR
END
```

نسخ محتوى مصفوفة إلى مصفوفة أخرى Copying Arrays

توفر بعض لغات البرمجة وظائف Methods خاصة بنسخ محتوى عناصر من مصفوفة إلى مصفوفة أخرى، فمثلا نجد في لغات الدوت نيت (الفيجوال بسيك دوت نيت، وسي شارب) وظيفة تسمى Clone، تقوم بنسخ محتوى مصفوفة إلى مصفوفة أخرى، كذلك في لغة الجافا توجد دالة بنفس الاسم Clone تقوم بهذا الدور.

بإمكاننا الاستغناء عن هذه الدالة والقيام بنسخ محتوى المصفوفة بواسطة تكرار بسيط ينقل عناصر المصفوفة الأصلية إلى المصفوفة المستهدفة كما تعرض لنا الخوارزمية التالية:

الشفرة بالعربية

```
البداية
    //الإعلان عن المصفوفتين الأصلية والهدف
    رقمي مصفوفة_رقمية_المصدر [5] ؛
    رقمي مصفوفة_رقمية_الهدف [5] ؛
    رقمي عداد ؛

    //تخزين القيم في عناصر المصفوفة حسب الرتبة
    مصفوفة_رقمية_المصدر [0] → 100؛
    مصفوفة_رقمية_المصدر [1] → 760؛
    مصفوفة_رقمية_المصدر [2] → 324؛
    مصفوفة_رقمية_المصدر [3] → 109؛
    مصفوفة_رقمية_المصدر [4] → 221؛

    //نسخ عناصر المصفوفة المصدر إلى المصفوفة الهدف
    لأجل عداد → 0 إلى 4
        مصفوفة_رقمية_الهدف [عداد] → مصفوفة_رقمية_المصدر[عداد]
    نهاية لأجل

النهاية
```

```
BEGIN
        //الإعلان عن المصفوفتين الأصلية والهدف
        Int SourceNumericArray[5];
        Int TargetNumericArray[5];

        //متغير من نوع رقمي للقيام بعملية تكرار تذهب من أول عنصر إلى آخر عنصر في المصفوفة
        Int Count;

        // تخزين القيم في عناصر المصفوفة حسب الرتبة
        SourceNumericArray[0] ← 100 ;
        SourceNumericArray[1] ← 760 ;
        SourceNumericArray[2] ← 324 ;
        SourceNumericArray[3] ← 109 ;
        SourceNumericArray[4] ← 221 ;


        //عرض كل عناصر المصفوفة
        For Count ← 0 TO 4
                TargetNumericArray[Count] ← SourceNumericArray[Count];
        END FOR
END
```

الخاتمة

تم بفضل الله وعونه الانتهاء **من كتاب** "البحر الشاسع لدخول الخوارزميات من بابها **الواسع**" على أمل أن أكون قد وفقت في شرح وتبسيط أسس التفكير البرمجي، وتجدر الإشارة إلى أن هذا الكتاب ماهو إلا باب لدخول عالم البرمجة وتليه خطوات عملية أخرى.

يمكنك تحميل باقي كتب السلسلة وغيرها لكي تتضلع أكثر في البرمجة، كما يمكنك أيضا أن تشترك في القناة على اليوتيوب لتستفيد من المحتوى المعرفي المرئي.

لكل شيء إذا ما تم نقصان، فإن وجدتم في طيات هذا الكتاب أخطاء لغوية أو تقنية أو لديكم ملاحظات واقتراحات لتحسين السلسلة فلا تترددوا بمراسلتنا عبر العناوين الالكترونية التالية:

mobarmijoun@gmail.com

how2progspace@gmail.com

وكذلك زيارتنا على موقع أكاديمية المبرمجين العرب:

www.mobarmjoun.com

ومتابعتنا عبر قناتنا على اليوتيوب وصفحتنا على الفيسبوك:

www.youtube.com/EssaadaniTV

www.facebook.com/EssaadaniPage